Design of Tool for the Optimization of Deck Area Assignments with Integration into Existing Naval Ship Design Programs

By
Damian G. Oslebo

Bachelor of Science in Systems Engineering
United States Naval Academy, 2006

Submitted to the Department of Mechanical Engineering
In Partial Fulfillment of the Requirements for the Degrees of

Naval Engineer

and

Master of Science in Electrical Engineering and Computer Science

at the
Massachusetts Institute of Technology
June 2014
© 2014 Damian G. Oslebo. All rights reserved.

The author hereby grants to MIT permission to reproduce and to distribute publicly paper and electronic copies of this thesis document in whole or in part in any medium now known or hereafter created.

Signature of Author_____
Damian G. Oslebo
Department of Mechanical Engineering
May 09, 2014

Certified by_____
Chryssostomos Chryssostomidis
Thesis Supervisor
Professor of Mechanical and Ocean Engineering

Second Reader_____
Fredo Durand
Co-Advisor
Professor of Electrical Engineering and Computer Science

Accepted by_____
David Hardt
Chairman, Committee for Graduate Students
Department of Mechanical Engineering

Accepted by_____
Leslie Kolodziejski
Chairman, Committee for Graduate Students
Department of Electrical Engineering and Computer Science

(THIS PAGE INTENTIONALLY LEFT BLANK)

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.
PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
	Master's Thesis	15 May 2011-15 Jun 2014

4. TITLE AND SUBTITLE
Design of Tool for the Optimization of Deck Area Assignments with Integration into Existing Naval Ship Design Programs

5a. CONTRACT NUMBER
N00244-09-G0014

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Damian G. Oslebo

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Massachusetts Institute of Technology

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943

10. SPONSOR/MONITOR'S ACRONYM(S)
NPS

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
Many tasks in the early stages of ship design are manual and repetitive processes. One such task is in the realm of deck area arrangements.Fortunately this type of problem, known as Bin Packing, is well understood in computer science. Meta-heuristic methods of obtaining near-optimal solutions in a finite timeframe exist that are reasonable enough for use. This thesis presents a ship design tool that pairs two of these meta-heuristic methods with naval ship architecture and LEAPS based projects. The output of the tool is the general arrangements drawings in a universal CAD format that would be the starting point for more detailed arrangements.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Carol Stoker
			UU	111	19b. TELEPHONE NUMBER *(Include area code)* (831) 656-3025

Design of Tool for the Optimization of Deck Area Assignments with Integration into Existing Naval Ship Design Programs

Damian G Oslebo

Submitted to the Departments of Mechanical Engineering and Electrical Engineering on May 09, 2014 in Partial Fulfillment of the Requirements for the Degrees of

Naval Engineer
and
Master of Science in Electrical Engineering and Computer Science

Abstract

Many tasks in the early stages of ship design are manual and repetitive processes. One such task is in the realm of deck area arrangements. The allocation and assignment of areas in early stage ship design involves tracking the difference of total ship area envelope and all required areas to be placed for habitability, mission support, and propulsion capability among many. The problem becomes more complex with the addition of constraints involving required separation zones between other areas, affinities for certain areas or deck levels, and compartment subdivision. The Leading Edge Architecture for Prototyping Systems (LEAPS) database structure output from the Advanced Ship and Submarine Evaluation Tool (ASSET) provides a ship envelope and a list of areas requiring assignment. However, with over a hundred different area categories to place in a subdivided ship hull of a large number of compartments each with their own preferences and constraints, this problem is categorized as Non-deterministic Polynomial-time hard (NP-hard). A complete solution to an NP-hard problem cannot be found in polynomial time, meaning that finding the optimal solution in the design space is not realistically feasible as the problem scales upwards in size.

Fortunately this type of problem, known as Bin Packing, is well understood in computer science. Meta-heuristic methods of obtaining near-optimal solutions in a finite timeframe exist that are reasonable enough for use. This thesis presents a ship design tool that pairs two of these meta-heuristic methods with naval ship architecture and LEAPS based projects. The approach is divided into three major steps: a ship volume balance, a ship area balance, and an area layout of the ship footprint. The output of the tool is the general arrangements drawings in a universal CAD format that would be the starting point for more detailed arrangements.

Thesis Supervisor: Chryssostomos Chryssostomidis
Title: Professor of Mechanical and Ocean Engineering

Thesis Supervisor: Fredo Durand
Title: Professor of Electrical Engineering and Computer Science

(THIS PAGE INTENTIONALLY LEFT BLANK)

Table of Contents

Abstract 3

Biographical Note 8

Acknowledgements 9

1. Introduction 10

Executive Summary 10

Motivation 11

Problem Statement 12

Intent 12

Discussion of Previous efforts 13

Metaheuristic Algorithms 15

2. Design 17

Approach 17

Volume Allocation 18

One Dimensional Assignments 19

Solution Encoding 20

Measuring Success with a Consolidating Utility Function 20

Genetic Algorithm 23

Simulated Annealing Algorithm 27

Two Dimensional Assignments 29

Passageway Templates 30

Greedy Filling Algorithm to Distribute Areas 31

3. Implementation 34

Programming Language selection and graphical user interface toolkit 34

C++ / C# Hybrid programming language 34

Windows Presentation Foundation (WPF) for presenting model views 35

User interface layout 36
Volume Allocations View 36
One Dimensional Area Assignment View 38
Two Dimensional Area Layout View 39
Utility Function and Algorithm Settings 41
Limitations 42
4. Application Examples 43
Basic test case of algorithms 43
Test Case Ship Description 43
Area Allocation Stage 44
Two Dimensional Area Layout 46
Application to an ASSET developed ship 46
Nominal Ship Description 47
Volume Allocations 47
One Dimensional Settings 47
Two Dimensional Settings 48
Results 49
5. Summary 53
Conclusions 53
Opportunities for Future Research 53
References 55
Appendix A: Area Allocation Spreadsheet (Test Case) 57
Appendix B: ASSET Generated Area Reports 61
Appendix C: Area Allocation Spreadsheet (Applied Ship Case) 73
Appendix D: Ship Arrangement Drawings (Applied Ship Case) 87

Figure 1: Utility Function Hierarchy 21
Figure 2: Sample Gaussian distribution curve 22
Figure 3: Genetic Algorithm Decision Tree 26
Figure 4: Simulated Annealing Decision Tree 29
Figure 5: Sample compartment sector subdivision 31
Figure 6: GFA allocation of areas with and without structure 32
Figure 7: Greedy filling algorithm decision tree 33
Figure 8: Volume Allocations View 37
Figure 9: One dimensional assignments view 39
Figure 10: Two Dimensional Area Layout View 40
Figure 11: Test case ship zonal breakdown 44
Figure 12: Deck 1 Arrangements 46
Figure 13: Deck 2 Arrangements 46
Figure 14: Inboard Profile 50
Figure 15: Deck 4 Arrangements 50
Figure 16: Deck 3 Arrangements 50
Figure 17: Deck 2 Arrangements 51
Figure 18: Deck 1 Arrangements 51
Figure 19: Deck 01 Arrangements 52
Figure 20: Deck 02 Arrangements 52

Biographical Note

Lieutenant Damian G. Oslebo was commissioned from the United States Naval Academy in 2006 where he earned a Bachelor of Science Degree in Systems Engineering.

Upon completion of Navy Nuclear Propulsion Training, Lieutenant Oslebo reported to the USS Nevada (SSBN 733). He arrived during the start of the USS Nevada's midlife Engineering Refueling Overhaul in March 2008 and stayed through the USS Nevada's return to strategic service in April 2011, completed on time and nearly two million dollars under budget. Lieutenant Oslebo served as the Main Propulsion Assistant, Damage Control Assistant, Assistant Engineer, and the Quality Assurance Officer while on board the USS Nevada (SSBN 733).

Following that assignment, he attended the Massachusetts Institute of Technology (MIT) for a three year tour to study in naval architecture and electrical engineering. His specific focus while at MIT is this thesis with an effort to learn to make user interfaces in the US Navy more useable. His follow on assignment to MIT is working under Space and Naval Warfare Systems Command **(**SPAWAR**)** in San Diego, California.

Acknowledgements

First and foremost I would like to thank my principal advisors at MIT and its Sea Grant office. These individuals include Dr. Julie Chalfant and Professor Chrys Chryssostomidis from the mechanical and ocean engineering department. I would also like to thank my cross department advisor in the computer science and artificial intelligence laboratory, Professor Fredo Durand. Without their input, the scope of this project would have been very limited.

I would also like to thank several people at Naval Surface Warfare Center (NSWC) Carderock division. Specifically, I would like to thank Robert Ames, Robert Dellsy, and Adrian McKenna for their technical support. Their help was immense and necessary for my ability to understand NSWC application programming libraries and framework.

I would also like to extend my thanks to the Electric Ship Research and Development Consortium (ESRDC). Specifically, I would like to thank Blake Langland at the University of South Carolina (USC). Some of this thesis's underlying work and support was intended for use by this group, and I appreciate their help, courtesy, and direction.

Finally I would like to thank my wife, Noon, for her support during this project. She was a saint the entire project often sacrificing her desires to support me. I can only hope to help half as much as she helped me during her time of need.

1. Introduction

Executive Summary

An essential task that is required for ship designs in the early concept stages is the production of general arrangements plans. Most ships have the same general or very similar placement of space areas throughout its decks. For example, the aviation equipment is usually placed aft and on or near the weatherdeck. Anchor support equipment is located near the foc'sle . Conning stations are located high in the ship in the bridge for the greatest line of sight. What usually follows the placement of the space in a particular compartment of the ship is a check on what else needs to be placed in that compartment and whether there is enough space to support other areas in the compartment.

These tasks of placing equipment in various spaces can actually be thought of as repetitive processes. This is because of the back and forth placement tracking of areas and subsequent reassignments after discovering some placement combinations do not work well. Repetitive processes are very good candidates for computer program development and optimization techniques. The process of placing spaces into compartments maps well to a computer science topic known as bin-packing. Bin-packing is categorized as a combinatorial non polynomial time hard problem (NP-hard). NP-hard problems require a full enumeration and search of the solution space that is usually not feasible on reasonable time scales. However, we are fortunate to have near optimal solutions to these types of problems using heuristics techniques such as genetic algorithms or simulated annealing.

This thesis develops a tool for area reservations, known simply as Ship Area Arrangements (SAA), to interface with projects developed in the Advanced Ship and Submarine Evaluation Tool (ASSET). SAA seeks to automate the repetitive tasks of area reservation and placement. The tool is implemented in three major stages. First, the volume allocation stage determines what spaces are available for arrangement on the ship. Operational tankage is assigned in this stage according to the stability, strength, and trim requirements of the vessel. The engineering compartments are assumed to be fixed in this stage by the required stack length of the shaft line. The second stage is the one dimensional process of placing space areas into compartments. It is one dimensional because only a comparison between the compartment's available area and allotted area is made. A utility function is paired with a genetic or simulated annealing algorithm to place areas based on the designers' desires and preferences during this stage. Finally, the third stage of the tool lays out the area in two dimensions for further work in Computer-Aided Design (CAD) programs. Passageway templates were used during

this stage to handle their distributive nature. A "Greedy filler" was used for the two dimensional distribution of spaces, where spaces of higher area allocation amounts are placed in priority over lower area allocation amounts.

This tool does not replace the need for further detailed design in CAD software tools. Additionally, the approach presented is not without flaws. One significant one being the limitation to account for the distributive nature of some systems, such as electrical distribution. However, I believe that it provides a very good starting point for detailed design.

Motivation

Naval architecture can be broken down into many often intertwining design and analysis categories. Some examples of these inter-related categories include ship stability analysis, intact strength analysis, and propulsion assessments, as well as the topic of this thesis: ship general arrangements. Design decisions made in one category often impact other design categories. For example, the selection of a hull plate thickness for a strength decision can and will impact the weight and stability results of the ship.

ASSET is a program developed by Naval Surface Warfare Center (NSWC) that manages the interaction of these competiting categories. This tool is the current standard for ship development and planning for United States Navy assets in the early stages of program conception. ASSET provides values for major space requirement definitions in an area spreadsheet report based on a number of factors including hull size, manning, and payload. However, ASSET is implemented in a tables and forms user interface format, which is non intuitive for use with spatial variables such as area and volume. Visualizing one dimensional numbers and measuring their impact to the design is often difficult, especially without a reference scale drawing. A better approach could show these spatial variables in two or three dimensions with respect to the hull volume and decks.

The generation of detailed plan drawings is necessary with all new-concept naval ships. This step is accomplished by using the ASSET-generated space reports for the required spaces, and then plotting these values inside the hull volume. One major reason why this step is often not easily accomplished is that spreadsheet data of one dimensional values does not easily translate naturally into two or three dimensions. Another reason for this step's complexity is that in order to perform general arrangements properly, multiple programs or accounting methods should be used simultaneously. Generally, this stage uses a CAD program doing the geometry calculations and area layouts, and a separate program manipulating the spreadsheet after each area is laid to ensure all areas are laid. This

process is made more tedious as the area spreadsheet typically has more than a hundred area assignments for naval vessels.

Problem Statement

The essential problem that this thesis tries to address is that the area placement process is more difficult than it should be given its tedious nature. This problem is compounded by the fact that no current solution exists that marries ASSET to other forms of CAD for detailed plan drawings.

Intent

The purpose of this tool is to provide a better bridge to CAD software from an ASSET developed design. As discussed in the problem statement, multiple programs must be used simultaneously to perform general arrangements from ASSET. This thesis offers a solution that intends to simplify the simultaneous operations into one program approach and speed up the tedious assignment of areas to the compartment hull volume.

A Solution That Simplifies Area Accounting and Spatial Awareness

This thesis intends to simplify the area accounting process by separating this process into two steps. In order to accelerate the process of area placement, metaheuristic methods are used to generate starting locations and placements for the required spaces into the hull's compartments in the first step. A greedy filling technique is then used to speed the process of laying the areas spatially in two dimensions in the second step. These areas can then be modified in the SAA tool or in external CAD programs.

A Solution That Bridges CAD Tool Interaction

This solution directly interfaces with ASSET without an extensive import process that is often required by other ship design programs that often do not accept ASSET's native database file format. Although an implementation detail, directly interfacing with ASSET was required by the SAA tool solution and eliminates a possible source of error that non-automated import indroduces. ASSET is the standard for early ship design of U.S. Navy ships; as such, program solutions for the Navy in early stage ship design should also accommodate its format for relevancy over solutions that do not.

The output should interface with other CAD software without its own conversion process prior to use. This program solution uses the standard drawing exchange format (.dxf extension) that is used in advanced CAD programs such as AutoCAD by Autodesk and Rhinoceros by McNeel North America.

Discussion of Previous efforts

There is a long history and progression of ship design tools. Attempts to integrate computer aided decisions into ship general arrangements have taken place for at least a half century. I present a brief sampling of these efforts to provide the reader a better understanding of the development of past tools. This better understanding of the past will also help the reader understand the design decisions going into the SAA undertaking.

Of the earliest attempts, one of the more notable programs was the U.S Navy development of the Computer Aided Ship General Arrangement Program (CASGAP) in the mid-1970s (Olson, 1998). It featured ten modules for ship design. Of the modules, the most important to the general arrangements designer was one that generated compartment requirements that essentially subdivided the vessel into "chunks". These chunks had a number of characteristics such as required area and volume, dimensions, global location in the ship, and even Ship Work Breakdown Structure (SWBS) numbers. However, due to the large number of independent variables in each chunk and available processing power, CASGAP did not make any attempts in optimization. Instead, designers would be left to construct the general arrangements "manually" and look for improvements by slight alterations in the design. A "manual" placement is one where there is no decision or suggestion by the program in the area's placement. The idea of subdividing the ship into compartment spaces in the SAA tool (as opposed to deck levels) can be rooted to this application.

The follow-up program to CASGAP was known as the General Arrangements Design System (GADS) developed in 1987 by NAVSEA (Olson, 1998). The design principles of GADS was very similar to CASGAP in that no actual decisions were made by the program. GADS big improvement over CASGAP was that its functionality was split over 29 different sub programs or modules that could be run independently of the overall program. The end result of this program subdivision was a marked increase in speed allowing the user to see real time changes in the design. This allows for better usability for the user to adapt to changes in situ. For example, the user can receive errors immediately after changing compartment bulkhead spacing when the required areas of the compartments become less than threshold values of the program. While I do not use sub modules, also known as "agents", to perform optimization tasks in the SAA program, I recognize this as an area for future work. Similar to GADs, wherever possible, I tried to display as much real time data as available to the user to aid in decision making.

Design approaches splinter in the early 1990s as processing power began to be able to handle more complex tasks. Several researchers (Cort, Hills, Nehrling) began development of applying fuzzy set

theory as a way to optimize general arrangements (Nick, 2008). Fuzzy set theory is a method of assigning a utility scale to designer preferences and combining these preferences into a single utility function to optimize. Fuzzy set logic was developed to numerically assign a value to a good design. SAA takes influence from this work by using a utility function to measure a good design according to user preferences.

Another design approach arose in the early 1990s from England for general arrangements. Andrews of University College London proposed using big building blocks to construct designs originally to submarines and later to surface ships (Andrews, 2006). The idea of building blocks is to mix and match different configurations or layouts in a design. For example choosing the best communications suite for the bridge out of a variety of options. In a building blocks design approach the ship becomes more modular; allowing the designer to choose different configurations to meet design requirements such as choosing between 20 MW or 40 MW engine room configurations for the same volume in the ship. The SAA does not incorporate modular building block configurations into general arrangements, but Andrew's work inspired the use of zones into the overall utility function.

In recent history, Eleanor Nicks of the University of Michigan in 2008 applied a genetic algorithm to a modified fuzzy set logic proposed by Nehrling for general arrangements (Nick, 2008). This research is now being implemented into the Intelligent Space Arrangements (ISA) program at Carderock. The implementation of the SAA genetic algorithm and its gene selection approach is from this research.

Finally, one other previous research effort to note, from Bart Van Oers of the Netherlands, proposed a three dimensional bin-packing approach using a genetic algorithm for general arrangements (Oers, 2011). His implementation is a mix between the building blocks approach of Andrews and the fuzzy set logic application of Nicks. His approach breaks down a ship into a voxelized (3D pixel) grid, and places large objects from a database of components into the space with a genetic algorithm. His approach was not incorporated into SAA for a couple of reasons. The first issue being the existence of the database information available to the user. As of now, there is no database of components or room configurations available from which to choose. Producing such a database is a course of study in itself. The second reason being the computational complexity of his implementation. To implement his approach would most likely require the use of multiple computers networked together to get results in a reasonable timeframe. For these reasons, his approach was not pursued. Ultimately, the SAA program is meant to be a bridge to other CAD environments for detailed design.

Metaheuristic Algorithms

Metaheuristic algorithms are a class of optimization methods used to search a solution space that is too large to enumerate and test each solution. "Metaheuristics are applied to *I know it when I see it* problems" (Luke, 2013). What this essentially means is that metaheuristics are used when one does not know what the form of the solution will take, but knows the correct answer or a near-optimal answer when the solution is presented. This is similar to a wine enthusiast or sommelier not knowing the exact composition to a good wine, however discerning a good wine from a bad wine due to the taste elements comprising it. Metaheuristics are used when the model behind a solution cannot be concretely defined as in the case with subjective evaluations. Area preferences and general arrangements, because they are subjectively evaluated, are good candidates for metaheuristic solution techniques.

Metaheuristic algorithms depend on some degree of randomness to search the solution space (Luke, 2013). Each metaheuristic algorithm has its own strengths and weaknesses, and types of solutions to which they are best applied. I present a sampling of some of the metaheuristics that can be used to solve subjectively evaluated problems. Some of the metaheuristic methods that be employed are a straight random search, hill climbing, simulated annealing, genetic algorithms, particle swarm optimization, and neural networks. This list is not inclusive of all metaheuristic algorithms, as this area of research is relatively new with some methods being introduced as late as the mid 1990s and more methods being developed currently under active research.

The alternative to metaheuristic algorithms is a gradient based optimization. Gradient based optimization is the same as using the Newtonian calculus methods of finding the global optimum. The slope and curvature of the function that describes the solution space is solved for and solutions that occur with a zero derivative are candidate maxima and minima. Solutions for multidimensional spaces are computed using Hessian matrices, which computes the derivatives and curvatures of the dimension to dimension coupling over the solution space.

Gradient based optimizations will find the guaranteed optimal solution but are not considered in this paper for one significant reason (Luke, 2013): to use a gradient optimization approach, one must know the function that describes the solution space. In the case of area placement, one does not know this function because the terms that describe the solution space have interactions between each other that cannot be modeled by any equation. The area interactions are subjective and complex such as in the case where an area can be placed into two compartments on the boat according to location preferences but the area can only fit into one compartment based on the compartment sizes. This problem scales upwards when we consider hundreds of space areas to place with each their own

preferences. Instead the better approach is to use metaheuristic algorithms and treat the function as a black box with inputs that the user can adjust such as location preference and compartment filling preference.

Simulated annealing, random search, and hill climbing belong to a class of metaheuristic algorithms known as single state methods. Single state methods evaluate only one solution at a time and there is no attempt to determine any relationships that states have between themselves. Random search is exactly as the name implies as it selects random solution points and selects the best solution that was generated. As it sounds this technique is very unlikely to find any results that could be feasible. Hill climbing method tries to combine the gradient based approach into a random search by altering solution states slightly and only accepting solutions that are better than the previous states. An evolution of the hill climbing approach is simulated annealing. Simulated annealing recognizes also inferior solutions initially according to a "heating and cooling schedule" when searching a design space (Ginneken, 1989). By considering less viable solutions they produce better solutions by avoiding local optima stagnation points in the design space. An advantage of using single state methods is their ease of implementation, with regard to code complexity, processing speed, and consumption of computer memory resources. The major disadvantage of single state methods is their search power. Search power is the ability to search the entire solution space. Single state methods are at a disadvantage because only one point in the design space is examined at a time.

Population methods overcome the disadvantage of search power at the expense of implementation complexity. They examine, as the name implies, a collection of possible solutions at once. Particle swarms and genetic algorithms are two ways of searching the solution space with a collection of solutions. Particle swarm optimization acts much like hill climbing method done in parallel with many possible solutions. The current and global best solution particles usually have the ability to interact and influence the solution trajectories of the other particle solutions. This is similar to a flock of geese following the goose in the front of the flock. Genetic algorithms employs a survival of the fittest processes to finding the best solution. For genetic algorithms, the best solution is a combination or breeding of the most viable solutions.

Neural networks lie outside the single state and population methods. The analogy to this solution is the human central nervous system. A network of nodes is set up much like the neurons in the brain are connected. The connections are initially set to random weighting and refined through a training process. A solution represents the weighted total sum output of this collection of nodes. Nodes in the case of area placements could be represented by each individual compartments in the hull and

the interactions between the nodes could be represented by the areas placed in each node. Neural networks were not considered for this thesis because of their complexity to implement and the fact that the training process severely limits the flexibility of the tool. The network would be different for each hull, because the number of compartments differ for any given ship. The user then would have to go through a training process with each arrangements process by selecting configurations that the user likes or dislikes. The neural network that results from this configuration process would be different for each ship and each user. And ultimately the user does not want to spend time training arrangement configurations that they could be spending adjusting the results of reasonable solutions.

For the design of the SAA tool, I chose to employ simulated annealing and genetic algorithms. I chose to employ two methods primarily to demonstrate that the SAA tool can have a plugin type ability that can be later modified with different solution algorithms from research and for debugging purposes. The reason I chose simulated annealing as a solution algorithm was that as a method it is slightly better at finding solutions than hill climbing and for its ease of implementation. Genetic algorithm was chosen for its proven results in prior research that was applied to general arrangements on ships (Nick, 2008).

2. Design

The design description portion of this thesis is broken down into four major sections. The first section explains the overall approach of the SAA and some of the reasons for a three stage approach. The following three sections explain some of the theory details behind the three stages described in the overview: Volume Allocation, One Dimensional Area Arrangements, and Two Dimensional Area Arrangements.

Approach

The SAA tool is divided into three major stages. The first stage is a volume assignment module that determines the compartment breakdown of the ship. At this stage, the user defines which compartments are tankage or arrangeable space as well as assigning the zonal breakdown of the ship. The next stage is a one dimensional assignment of space areas into the arrangeable compartments of the ship. The result of this stage is a balanced spreadsheet that considers compartment area available against what is required area by the spaces. This stage can be accomplished with a computer heuristic algorithms or with manual spreadsheet entry. The last stage is drawing the two dimensional footprints in the compartments for export to CAD format. There is also a balance in this stage between how much area is physically laid in the compartment to the value set in the second stage of the SAA tool. This stage can be accomplished by using the greedy fill algorithm or by manual drawing.

The SAA tool approach is divided into three stages for a couple of reasons. On the program workload side, the three stage approach divides the problem statement into three smaller, more manageable problems. Optimizing a three-dimensional problem of arrangements becomes a much larger problem set for the program to work on. Reducing the solution space to one dimension in the second stage still produces desirable ship configurations in much faster time. For perspective, consider the one dimensional case of a ship with 50 compartments and 120 space areas to place. If we only consider the cases where we fully place one area into any compartment, the solution space to enumerate is still 120^{50} or 3.5×10^{145} different unconstrained possibilities!

From a designer's perspective, having three stages allows for operator intervention between each stage. There may be instances where the utility function makes choices that the designer decides are unacceptable. For example, the utility function may decide the best place for the medical space is at the bow due to the interplay between factors in the utility function. The SAA approach allows for manual tuning of the computer generated solution while still offering similar speeds of automated design. Although there may be an encoded best solution described by a fitness number to a ship design, there is still an artistic aspect to general arrangements design that can only be seen from a modularized approach.

Volume Allocation

The first stage of the SAA tool begins with a volume allocation. This stage begins with an import of the ship from a Leading Edge Architecture for Prototyping Systems (LEAPS) database file. This file is the output of a ship design in ASSET. From the LEAPS ship project file, SAA imports the ship geometry and ship space classification system (SSCS) numbers. SSCS numbers are a U.S. Navy developed system that describe the volume and area requirements of the ship (NAVSEA, 1984). SAA tool also imports the compartment volumes and attributes. The starting state of the SAA tool is a set of compartment volumes and a set of required areas and volumes to place in the ship. The outputs of this stage are a reduced set of arrangeable compartments to place required areas and a volume balance spreadsheet describing compartment assignments.

The SAA tool makes a few assumptions after import to preserve the ship design. First, the SAA tool assumes compartment sizes remain constant. As a result, deck and bulkhead placements remain fixed. The reason for this is primarily to preserve the source file format as strongly suggested by the designers of the LEAPS format. Changing physical ship geometry has cascading consequences primarily because of how LEAPS file format saves surface intersections. An example would be to resize a

compartment for more space, however this change would not only affect the mesh structure of compartment resized but also neighboring compartments, the hull, and any major ship components contained in the affected volumes (i.e. exhaust stacks). The LEAPS format designers recommend returning to ASSET with all ship geometry changes to reproduce a new hull and compartmentation for other programs to properly use. Next, engine room locations are also assumed to be fixed for the same reason. Additionally, the stack length of the shaft line is already deemed adequate at this stage of the design. Lastly, exhaust and intake stacks are also fixed, but are not theoretically required to be fixed. The design decision to keep the stacks fixed was to reduce program complexity and is an area recommended for further work.

At this stage there is a volume balance that must be achieved by considering the required tankage volumes, ship volume, and the remaining compartment available area. Ship designs with less area available to arrange than what is required by SSCS areas should not proceed to the next stage. Ideally, a user of this tool would already have determined adequate tankage configuration from ship stability, trim, and strength analysis. The task then becomes matching the pre-established configuration from those analyses to the ship project.

Finally, the notion of compartment zonal assignments is introduced in this stage. Zones in the context of the SAA tool are compartment groupings. This functionality allows the user to specify adjacency preferences to group areas together. It is not an SAA requirement that compartments be contiguous to be in the same zone. This may aid a user in placing an area by the heuristic algorithms in a situation where there may be an area that could go in one or two places but the user doesn't really care where exactly. An example of this could be setting up a berthing zone throughout the ship consisting of several non-contiguous spaces; the user desires that all berthing be located within these spaces, but has no strong preference on the exact location within the berthing zone of Chief Petty Officer (CPO) quarters. Currently, there is no support to map multiple zones to one compartment. Mapping multiple zones to one compartment would allow for overlapping zones that exist in the ship, such as a compartment having an electrical distribution zone and a different HVAC distribution zone. This is desired functionality that is an area for further work for the SAA tool.

One Dimensional Assignments

The second stage of the SAA tool is the assignment of required areas to compartments. This process is accomplished by way of a spreadsheet balance between the available space within the arrangeable compartments decided in the first stage and the required SSCS areas determined by ASSET. This stage

can be accomplished manually or automated by a heuristic algorithm. Two heuristic methods were integrated into the SAA tool (genetic algorithm and simulated annealing algorithm) and are described in detail below. Before that discussion on the algorithms, what is meant by a solution and what is meant by a "good" solution are explained in the solution encoding and utility function sections.

Solution Encoding

In order to apply optimization techniques to a problem, a major hurdle to overcome is describing a solution state in the system. The solution state needs to be described in a way that lends itself to a full enumeration of all the possible states of the system. A solution state in the SAA tool is described as an array of integers; the size of the array equal to the number of areas to be placed. Each integer element in the array is coded to a compartment assignment of the SSCS required area. The entire area is then placed into the compartment where the solution state describes it to be. An example state description is shown below, where there are four SSCS areas to be placed and x1 through x4 variables represent integer assignments to compartments on the ship:

Ex: $[x1|x2|x3|x4]$

One problem with this type of solution encoding is handling distributed systems throughout the ship. Each required area is placed entirely in one compartment. Problems arise when placing distributed systems, for example, placing all the SSCS required area allocated for passageways (SSCS 3.8) into a single compartment does not make sense. In this version of the SAA tool, SSCS required areas are not broken down into partial blocks because of added complexity to describe a state in the system. The state array length between two different states is always the same length: the number of SSCS required areas to be placed. Consider the following: *Into how many subdivisions can a SSCS required area be broken? Are the broken down entries equally divided amongst each other? How do you describe the geometric center of these split individual areas for global placement optimization? How do you compare one state array to another state array if they have different lengths?* These questions make it difficult to describe a solution space. To handle the problem of distributed systems there is an option in the SAA tool to skip the automatic placement of user-chosen distributed systems. There may be potential for breaking down SSCS numbers found in recent work in genetic algorithms with variable length genetic chromosomes to encode a more flexible solution state; this is an area for future research.

Measuring Success with a Consolidating Utility Function

In order to run an optimization program there needs to be a way to define or quantify success. For general arrangements, multiple independent factors play a role in determining a good design and there

needs to be a way to consolidate user desires. Therefore, the SAA tool implements the decision-matrix method, otherwise known as the Pugh method, to quantify solution fitness (D. Frey, 2009). The decision-matrix method applies a weighting value to various aspects of a solution and the resulting weighted values are summed to produce an overall utility. Weighting values are assigned by the user according to how important each aspect is to the current design.

To accommodate more flexibility in utility calculations, there is also an ability to consolidate utility functions by taking the minimum of the comprising utility functions to produce an overall utility value. This may be desirable in some circumstances where the user considers all low utility values as unacceptable even if higher priority (higher weighted) items meet acceptable values to the user. The factors determining overall ship fitness are shown in a hierarchical structure in Figure 1.

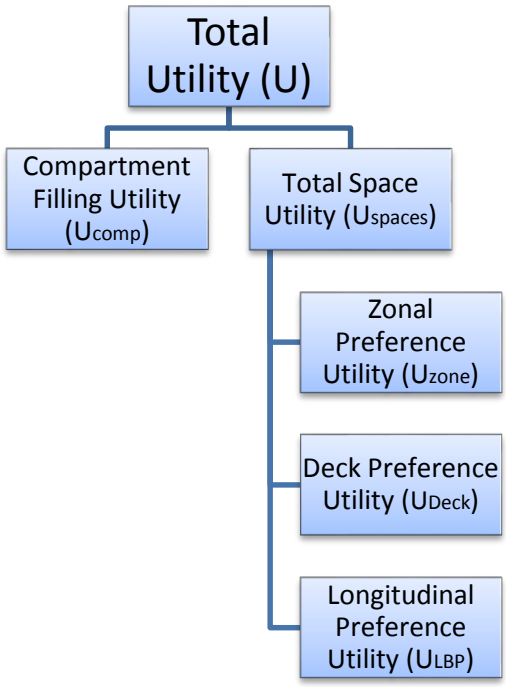


Figure 1: Utility Function Hierarchy

The top two factors determining overall utility of a solution (*U*) are compartment filling (U_{comp}) and total space utility (U_{spaces}). Weighting factors (*wf*) for each utility can be set by the user. By default, the utilities are equally weighted; thus, wf_{comp} is equal to wf_{space} unless changed by the user.

$$Total\ Utility(U) = \left(wf_{comp}\right) * \left(U_{comp}\right) + \left(wf_{space}\right) * \left(U_{spaces}\right)$$

Alternatively, if utility calculation setting set to minimum instead of average:

$$Total\ Utility(U) = Min(U_{Comp}, U_{Spaces})$$

By default, the compartment filling utility value (U_{comp}) is calculated by taking the average of all the individual compartment filling utilities. To determine an individual compartment's filling utility, SAA calculates the individual compartment's filling percentage by taking how much area is assigned to the compartment over how much area is available to the compartment. A user-defined Gaussian distribution centered about a user-specified desired filling percentage mean value is then applied to the filling percentage for each compartment to determine the weighting factor for that compartment. The user specified filling percentage should not be set too high to account for passageway space not considered in the utility calculations. The user can specify the falloff of both tails of the Gaussian distribution. Over-filling of a compartment is undesirable because there is not enough physical space to fit all the areas into the space. Under-filling a compartment is not desirable because not only does it result in the underutilization of the compartment, it also leads to capacity-limited compartments elsewhere on the ship.

Figure 2 shows an example Gaussian distribution that can be used to determine compartment filling utility from a compartment fill percentage. The falloff to the right of the mean is sharper than the left to show that the penalty for overfilling is more severe than for under filling, and the mean is set at 90% to account for passageway margins.

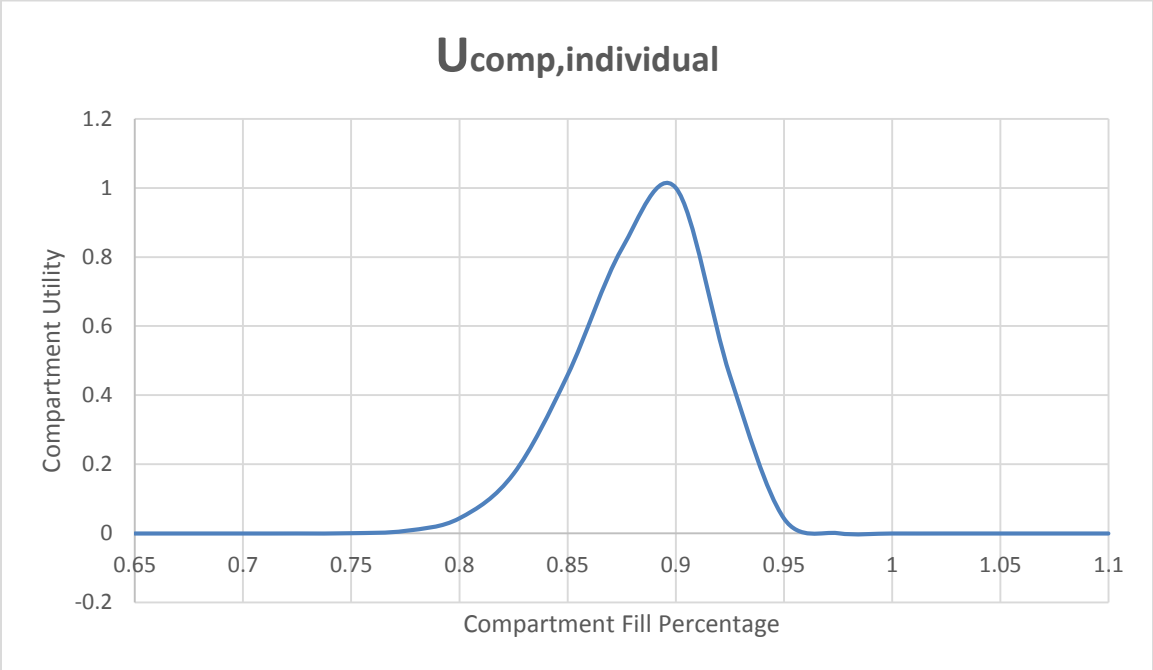


Figure 2: Sample Gaussian distribution curve

The space utility is comprised of three other utility measures representing the average space satisfaction of individual SSCS assignments according to user preferences. The three utility measures composing the individual space utility (U_{space}) are longitudinal preference utility (U_{LBP}), zonal preference utility (U_{zone}), and deck preference utility (U_{Deck}). To calculate a space utility, the SSCS required area's centroid is determined by its placement in a compartment. For the longitudinal and deck preference utilities, the required area's difference between the centroid and desired location is taken as percentage of ship's length between perpendiculars or ship's height respectively. This difference is applied to a user specified Gaussian distribution to recover a utility value similar to how the compartment filling utility is recovered above. The max value of these utilities occurs when the space area centroid is on the respective deck or at the specified longitudinal location. Zonal preference utility is binary because zones can be user distributed throughout the ship and not necessarily geometrically co-located. This means zonal utility is at full value of one or zero depending if the required area centroid is in a desired zone or not.

$$U_{space} = (wf_{zone}) * U_{zone} + (wf_{LBP}) * U_{LBP} + (wf_{Deck}) * U_{Deck}$$

Alternatively, if utility calculation setting set to minimum instead of average:

$$U_{space} = Min(U_{LBP}, U_{Zone}, U_{Deck})$$

Genetic Algorithm

A genetic algorithm was implemented for the optimization of the overall utility function. Genetic algorithms are in a class of optimization methods known as heuristics optimizations (Luke, 2013). Heuristic methods allow a smart and efficient search of the solution space on time scales well below full enumeration of solution states. A very good solution can be recovered from the state described by the genetic algorithm's elite chromosome or highest ranked solution. This solution is not the guaranteed optimum of the solution space. In the case of mapping numbers to preferences, which are not necessarily exact to begin with, the solution can be considered good enough.

The general principle of how a genetic algorithm works is based on natural selection and survival of the fittest. Initially a large population of randomly or procedurally generated chromosomal descriptions of solution states are seeded. The fitness of each chromosome of the population is evaluated, and the population is sorted in order of fitness/utility value. Then a sampling of the top performing chromosomes of the current population is taken to reseed a new population. Chromosomes that were not selected in the previous gene pool are removed from the gene pool. The process repeats on evaluating and repopulating solution states until a user specified stopping point is reached.

The genetic algorithm implemented in SAA is very similar to the genetic algorithm from Eleanor Nick's doctoral thesis (Nick, 2008). Like the genes that comprise the genetic algorithms, not all genetic implementations are alike. Genetic algorithms can differ from each other in a few ways. A few notable differences of the SAA tool and Nick's implementation to other genetic algorithms are how it performs gene selection from the current pool, what operations are performed on the gene pool to produce the next generation, and the stopping conditions. These differences are described in more detail below.

Tournament selection

One of the ways the SAA genetic algorithm is unique is how it selects a gene pool for the next generation. The SAA tool uses a two stage tournament selector in the selection process of generating a gene pool. The purpose of the two stages is to allow for survival of the fittest selection to occur while allowing for a small chance of selection of some of the weaker solutions. Weaker solutions manipulating/breeding with more elite solutions may result in better solutions than breeding two elite solutions together. Empirical evidence showing a comparison between a purely survival of the fittest selection process against this more balanced selection process can be found in (Nick, 2008).

The first stage of the tournament selector randomly samples the current chromosomal solution pool to generate a smaller chromosome pool size for the second stage. This means that before a survival of the fittest selection process occurs in the second stage, some of the better solutions may be randomly excluded from the next generation; however, the elite (top-scoring) chromosome solution is always retained in the population pool.

The second stage begins with sorting the tournament results of the first stage by order of utility. A random selection is made from the sorted pool's top three choices, with the selection moving into the gene pool of the next generation. The tournament begins again at the first stage and the process repeats until a gene pool matching the same size of the original population is selected. With this type of selection process, there is high potential for multiple copies of some of the same chromosomes in the pool for the next operation.

Chromosomal operations

Another one of the ways genetic algorithms vary is in the operations performed on the gene pool to produce the next generation. The solution space for general arrangement configuration certainly has many local optimums. Chromosomal populations will want to stagnate around these local optimums without intervention. Chromosomal operations allow the solution space to be walked intelligently and reach beyond these local optimums into other regions of the solution space. There are four

chromosomal operations performed on each generation before being evaluated to seed the next generations.

The first chromosomal operation performed is known as simulated binary crossover (SBX). SBX is a uniform crossover operation, which means that it operates on all genes (*the array-indexed elements of a solution encoded state*) of two parent chromosomes to meld two new daughter combinations. Simulated in the identifier SBX is based on the fact that the genes are integer encoded and not binary encoded. This operation concentrates on diversifying a population's chromosomal set and is an operation observed to occur in natural genetics. SBX works best when two parent chromosomes are dissimilar from each other. SBX loses search power over time, because gene pools eventually become very similar as a good solution is approached. As a result, SBX operations are performed a predetermined number of times based on the original population size to avoid wasting process time.

The second chromosomal operation performed is mutation. This operation is also seen in natural genetics and describes the random change of one gene to another random value. As in nature, the mutation rate is set to a low percentage chance of selection. Mutation search power is more significant the longer the simulation runs because of its low selection rate. Mutation acts to fine tune the optimal solution once located.

Single point crossover is the third operation performed on a chromosome population. This operation essentially swaps the head and tail of two chromosomes at a random point in the solution arrays. This operation, like the SBX, is effective in searching a solution space early on by imposing drastic changes in the gene pool when early solutions are poor.

The last chromosomal operation is known as swap operation. This operation has no biologic analogy, but Nick found the operation effective along with the mutation operation to increase the fitness of mature solutions. Essentially the swap operation takes the values of two random genes (indices) of a chromosome and swaps them so that the first gene has the original value of the second and the second has the original value of the first. This operation is best performed on indices standing for space areas similar in size.

Stopping conditions and reseeding

The last unique aspect of the SAA tool genetic algorithm is how it handles stopping and reseeding conditions. Periodically, the gene pool will stagnate as all the solution chromosomes become similar. Reseeding allows new life to be injected in the search space and new solutions to be explored. After a user defined set point is reached where the elite chromosome has not changed, the gene population is reseeded. All chromosomes in the gene pool are purged with the exception of the elite chromosome,

and the gene pool is reseeded with new randomly generated chromosomes. The stopping conditions for the genetic algorithm are reached whenever it has finished a predetermined number of runs or when it has been reseeded a user defined threshold amount of times with no change in the best solution. Limits on the numbers of reseeds allow the algorithm to run faster with safe assurance that a good general arrangements solution has been found.

Genetic algorithm decision tree

A decision tree is shown in Figure 3 that outlines the SAA tool genetic algorithm. This optimization process is run when the genetic algorithm is chosen as the space optimizer.

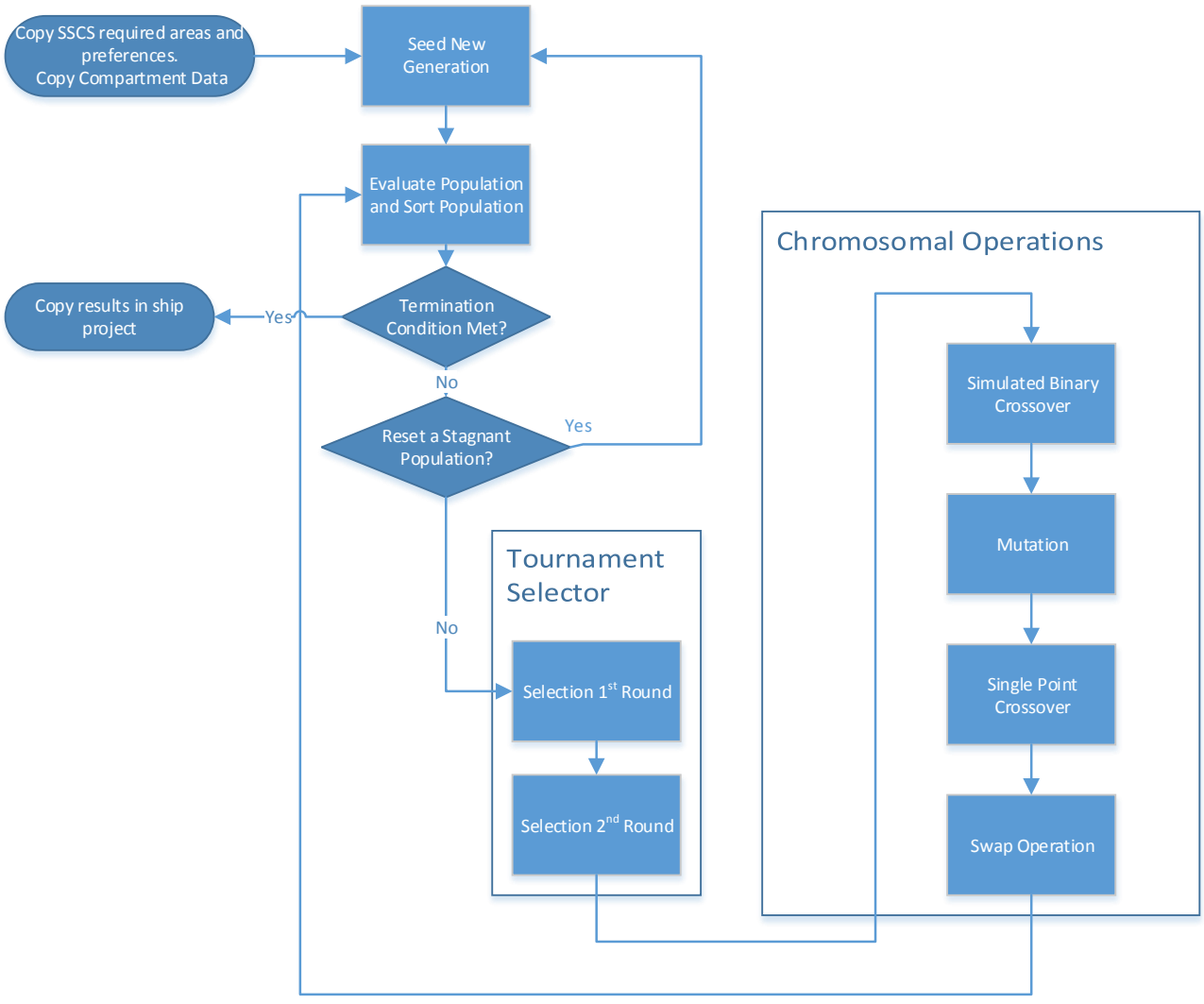


Figure 3: Genetic Algorithm Decision Tree

Simulated Annealing Algorithm

A simulated annealing algorithm was also implemented as an alternative optimization technique to the genetic algorithm described above. The general principle of how a simulated annealing algorithm works is based on intelligently moving up a solution gradient from a solution starting state (Ginneken, 1989). Its foundational principles are rooted in metallurgy. A solution is essentially treated with a heating and cooling schedule that results in a stronger solution in the end similar to a piece of worked steel. Simulated annealing starts with a randomly generated solution state that is set as the current solution state and the best saved solution state. The process then enters a loop until it reaches a stopping point set by a predetermined number of runs. The loop begins with a calculation of the solution's utility value and an acceptance probability, P(T), based on the cooling schedule. A neighbor state is then generated that is based off the current solution. For this optimization process, a neighbor is defined as a solution containing all the same elements as the current solution with the exception of one array indexed value in the solution being altered to a new compartment value. This is similar to the mutation operation discussed in the genetic algorithm section occurring every process iteration. If the neighbor state results in a better solution than the current solution, the neighbor solution is chosen as the current solution. If the neighbor solution is a less viable solution, the acceptance probability, P(T), determines whether the new solution is selected or not. At the start of the process, the acceptance probability starts high akin to a heated piece of steel. The algorithm is more likely to accept less viable solutions early on. However, as iterations of the algorithm proceed, the P(T) essentially cools and less viable solutions are less likely to be accepted.

Some of the unique aspects of the SAA tool implementation over a traditional implementation of simulated annealing are its best solution tracking and its use of a single heating and cooling cycle. Simulated annealing algorithms sometimes undergo alternating heating and cooling cycles as the process is run to help drive the solution out of local optima. This requires a fine-tuning of the algorithm tailored specifically to the problem and the gains in utility may not be worth the effort. The SAA Tool runs only one heating and cooling cycle, but the results seem satisfactory given the heuristic nature of the problem. In the traditional sense, the best solution over time is not kept and the output of the program is the current selected solution once termination conditions are met. The SAA tool keeps track of the best solution in case the current solution has veered away from an optimal solution due to the acceptance of a less viable solution.

Two Algorithms for one optimization

The SAA tool does not need to include two different optimization routines for area assignment. The reason for the development of the simulated annealing algorithm was initially to troubleshoot the results of the genetic algorithm. Simultaneous development of both algorithms made the process of finding errors in the implementation code easier to locate. It allowed me to modularize the algorithm and isolate working sections of both algorithms from ones with errors. It also provides the user the freedom and flexibility to perform one dimensional arrangements in one of three ways: manually, by genetic algorithm, or by simulated annealing.

From the test cases that were initially run, generally the genetic algorithm produced better results with higher utility values than the simulated annealing results. However, it is difficult to quantitatively compare the two algorithms, because they operate on two different time scales. The simulated annealing algorithm runs more iterations than the genetic algorithm to achieve similar results but uses less memory and runs faster. The difference in speed between the two algorithms is most likely attributed to two causes. First, there is a less complex process chain in the simulated annealing algorithm. And second, there are probably memory cache misses occurring from the large amount of memory accessed to store solution sets depending on the size of the chromosome pool each generation (Denning, 1968). In summary, simulated annealing runs faster than the genetic algorithm but is less accurate.

Simulated annealing decision tree

A decision tree is in Figure 4 that outlines the SAA tool simulated annealing algorithm. This optimization process is run when the simulated annealing is chosen as the space optimizer.

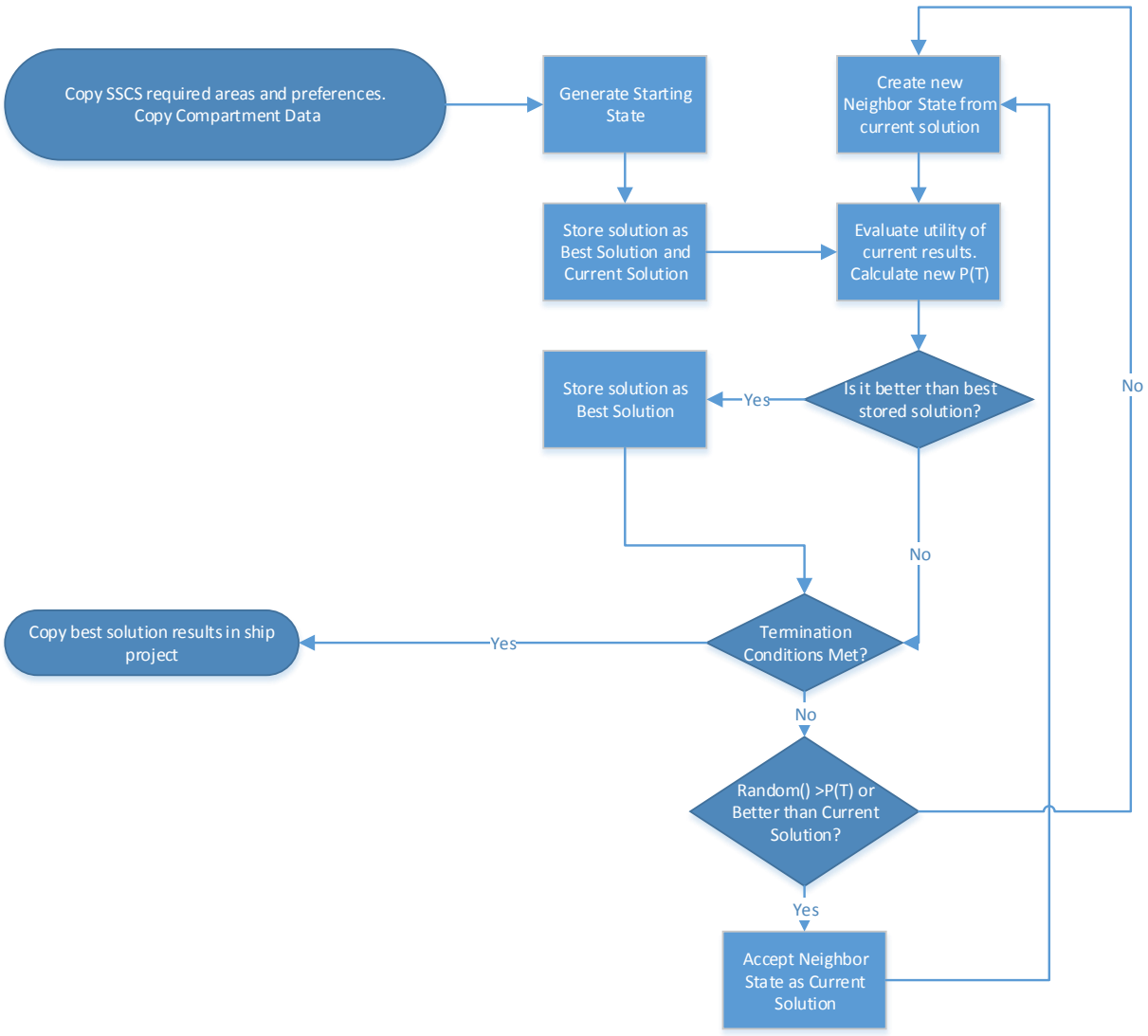

Copy SSCS required areas and preferences. Copy Compartment Data
Generate Starting State
Create new Neighbor State from current solution
Store solution as Best Solution and Current Solution
Evaluate utility of current results. Calculate new P(T)
Store solution as Best Solution
Yes
Is it better than best stored solution?
No
No
Copy best solution results in ship project
Yes
Termination Conditions Met?
No
Random() >P(T) or Better than Current Solution?
Yes
Accept Neighbor State as Current Solution

Figure 4: Simulated Annealing Decision Tree

Two Dimensional Assignments

Once all the space areas have been assigned to compartments the next stage is to lay these assigned areas into compartment two dimensional space. In the previous stage, the SAA tool accounted for passageways by including a margin in the mean fill percentage value for the compartments in the utility function calculation. Before the areas of the ship can be placed within the compartments, the passageway layout of the ship must be considered. One dimensional arrangements were made at a compartment container level, at a localized level. Damage control (DC) passages should know information beyond this compartment level, such as which compartments are adjacent to each other, to allow a continuous passage between compartments that ensures appropriate casualty response. The

SAA tool tries to take a global viewpoint approach by setting passage templates for compartments. Once the passageway layout is established, a designer can use a manual approach to lay out the assigned areas in the compartments or use the greedy filling algorithm to get a starting point into detailed design.

Passageway Templates

The SAA tool uses templates to handle the distributed nature of passages on the ship. Passages that dead end quickly do not serve their intended purpose of connecting spaces on the ship. The SAA tool allows for the manual placement of individual templates, or the use of a simplistic automatic generative function to lay passages on the ship in a logical fashion.

A template is basically a set of properties belonging to the passage in a compartment. Some of the properties contained in a template object are the passage widths, how many passages exist, and the connection points of the passage to other compartments. These properties exist to allow the SAA tool to properly draw the compartment template. Several template designs exist to aid in manual placement or edit. The basic types are a single passageway, a double passageway, and conversion pieces from a single to double or vice versa.

Users can use the SAA tool to lay passages automatically in one of two configurations. The first option allows for a single passageway running centerline of the ship, and the second option allows for a double passage to be run along the port and starboard sides of the vessel. After a DC deck is set by the user, the SAA tool runs two program passes on each one of the decks above the DC deck to set passages. Compartments below the DC deck are assumed by the tool to be watertight and there are no connections laid between compartments on those decks by the SAA tool. The first program pass on each deck lays the basic configuration of the respective template and also determines the connection points between compartments for the second program pass. Connection points are adjusted in some cases. For example when longitudinally placed compartments exist, the SAA tool will split a single passage into a double passage to allow transit in the longitudinal compartments. The second program pass then goes back and readjusts the passages to account for the changes in connection points.

A remaining issue left to handle for passage concerns is vertical transit between decks. Planning stairwells is possible in the SAA tool but is a manual process. One way to incorporate vertical transit in the automatic passage generator would be a separate third program pass on each deck to ensure stairwells line up between decks.

Greedy Filling Algorithm to Distribute Areas

Two dimensional placement of assigned areas is can be accomplished manually with polygonal shape editing tools in the SAA tool or automatically by using the greedy filling algorithm (GFA).

The GFA works on the principle that larger areas are more important than the smaller areas. The process loops through each arrangeable ship compartment and divides the available compartment area into sectors. Sectors are determined from several sources such as the passageway layout and obstructions such as intakes and exhausts. Sectors are also generated if a user manually places an area within the compartment before the GFA is run. Placing an area may be desirable in some cases such as placing a stairwell at a specified location in the compartment. This sector list is then sorted by area size for the next portion of the algorithm. Figure 5 shows a sample compartment to demonstrate sector subdivision. A single passage splits the compartment into two halves while the obstruction in the top half generates eight sectors available for area placement.

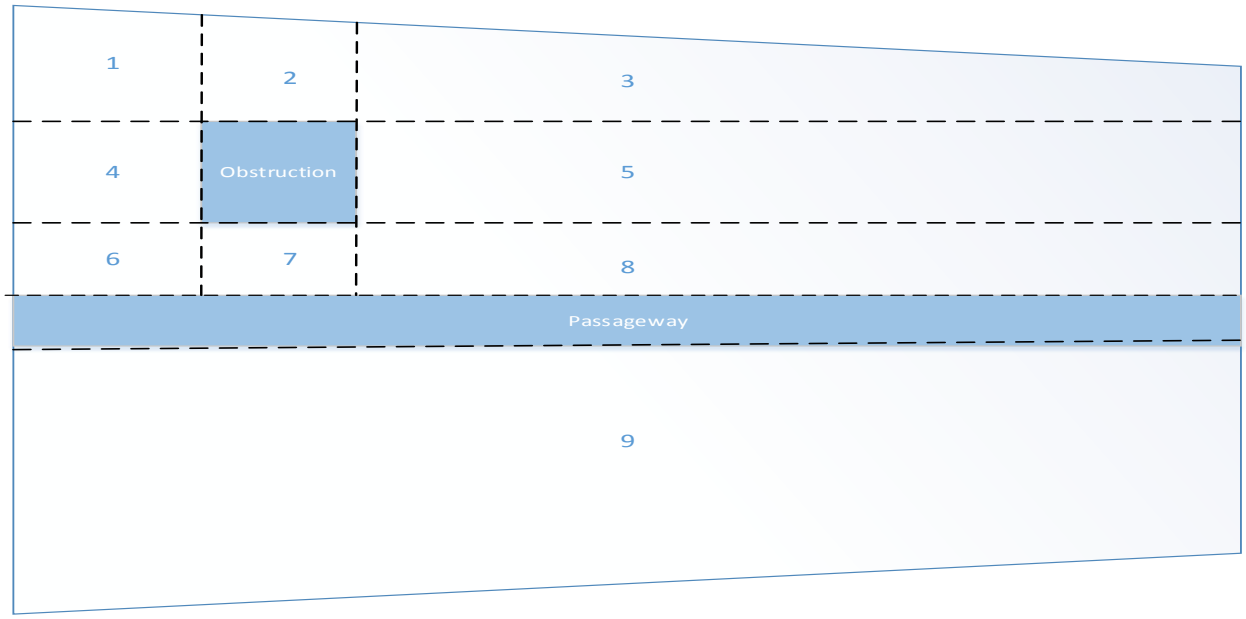


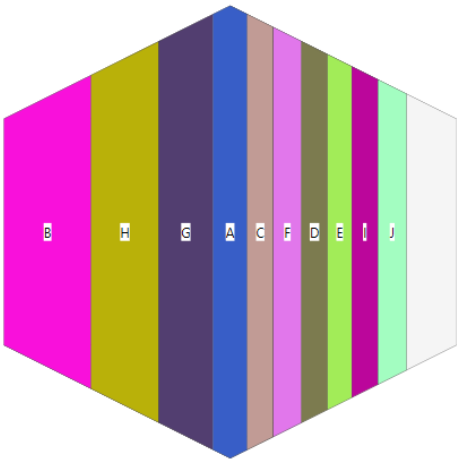
Figure 5: Sample compartment sector subdivision

After the sector list has been sorted by available size, a queue is set up of the required areas to be placed in the compartment by order of their size requirements. The GFA then enters a loop that draws areas from the queue until the queue has been exhausted or there are no sectors with available area left. The largest sector that can fit the area is selected for the area to be drawn. The area is split into two pieces if there is insufficient space in the largest sector with the second required area returning to the queue to process later. An area is drawn by starting at the boundary of the sector and advancing

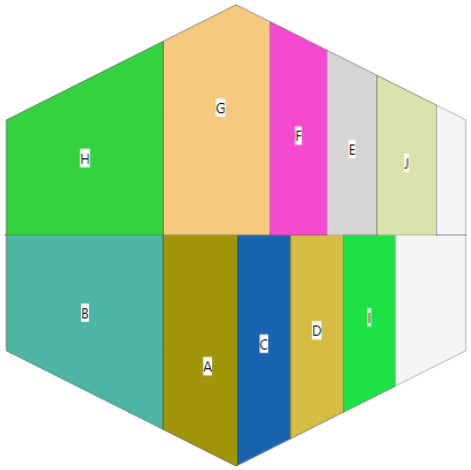
a marker until the required area amount is met. Once the area is drawn the sector is now smaller in size, so its area is recalculated and the sector list is resorted by available area again.

Because of the way the GFA uses the sector boundary to generate areas, small areas placed by the GFA may have small minimum dimensions in the x direction. This is remedied by the ability to use the manual polygon editing tools in the SAA program to resize these smaller areas for more useful design dimensions. Also, if the GFA is processing on the last assigned area of a compartment, it will automatically fill the remainder of a compartment sector with the last area to help minimize this consequence.

One result that occurs with the GFA is if the compartment has no structure, in the case of a compartment with no obstructions, is that the compartment will fill with areas that may not have the best aspect ratios for their respective areas. Figure 6 shows a sample compartment with generic areas drawn before and after a passage of zero width is introduced to provide a structure to the compartment. Both compartment drawings have been subdivided into the same size and number of areas but the aspect ratios of the 0 m passage compartment are better suited to fit equipment inside.



GFA allocation of areas in compartment without structure



GFA allocation of the same areas in compartment with 0 m wide passageway

Figure 6: GFA allocation of areas with and without structure

Greedy Filling Algorithm decision tree

The decision tree the SAA tool uses if the GFA is chosen to lay out assigned compartment areas is shown in Figure 7.

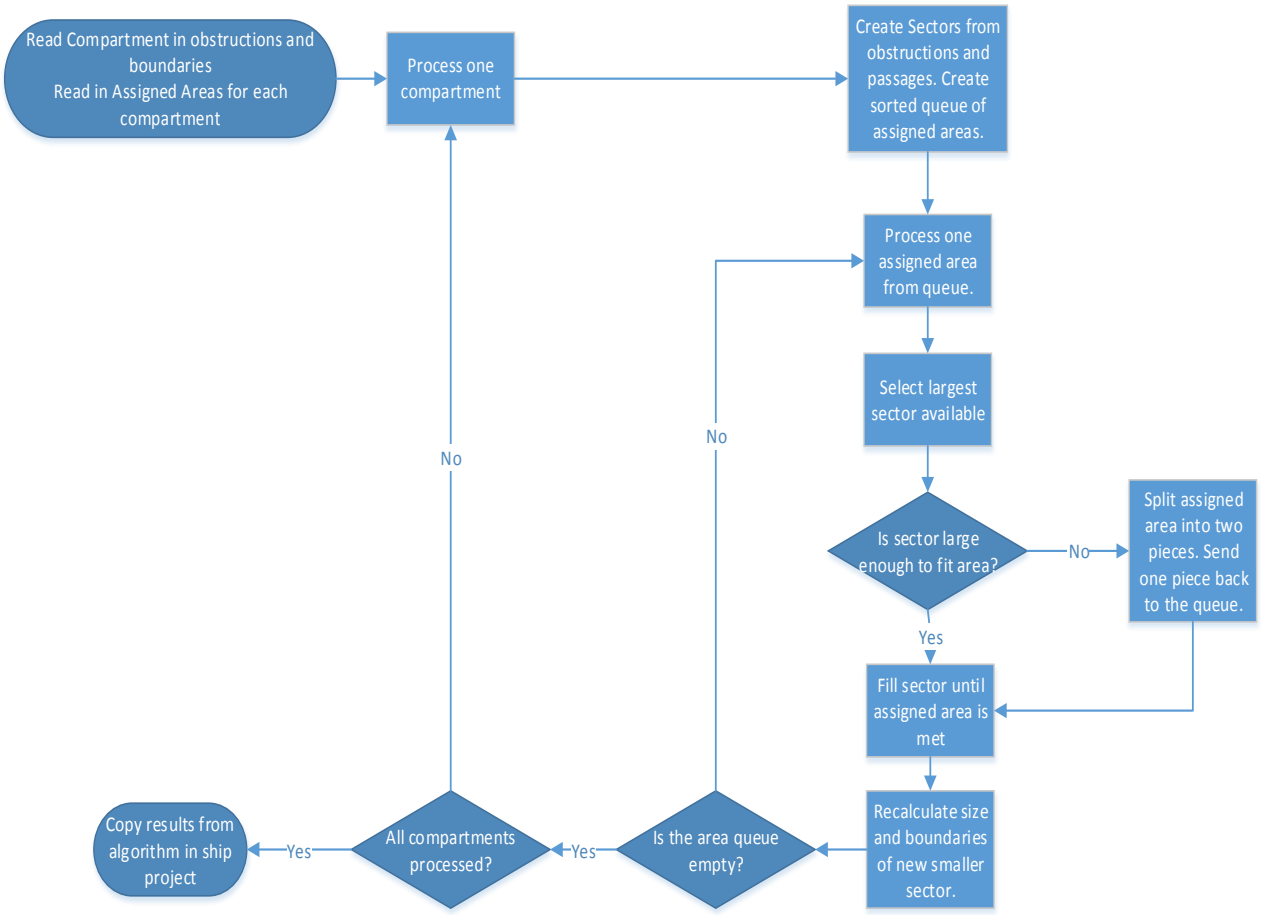

Read Compartment in obstructions and boundaries
Read in Assigned Areas for each compartment
Process one compartment
Create Sectors from obstructions and passages. Create sorted queue of assigned areas.
Process one assigned area from queue.
Select largest sector available
Is sector large enough to fit area?
No
Split assigned area into two pieces. Send one piece back to the queue.
Yes
Fill sector until assigned area is met
Recalculate size and boundaries of new smaller sector.
Is the area queue empty?
No
Yes
All compartments processed?
No
Yes
Copy results from algorithm in ship project

Figure 7: Greedy filling algorithm decision tree

The Greedy Filling Algorithm is not an optimization

It is important to note that GFA is not an optimization routine for a couple of reasons. The first reason is that there is difficulty defining the function to optimize when laying two dimensional objects. Nick proposes several criteria that could be used to evaluate a good layout such as area aspect ratio, adjacency requirements, and minimum dimensions in the x and y. The issue with defining a utility function similar to hers is that it depends on the technique that is used to lay the area down. Without going into detail to describe her method, it did not extend well into non-orthogonal ship boundary conditions. The conversion of her method to non-orthogonal conditions however is a recommend area for future work.

The second reason to consider is how long the optimization process would take. Ships can contain upwards of a hundred different compartments if the ship has longitudinally divided compartments. If an optimization routine on each compartment required three minutes to complete, it would take on the

order of three hours to serially process all compartments. The use of network agents to run optimization methods efficiently in parallel is being considered for Carderock's ISA program (Anthony Daniels, 2010).

3. Implementation

The implementation of the SAA tool is broken down into four sections. The first section describes the language implementation and presentation framework toolkit to develop the SAA tool. This section is written primarily for those individuals who are interested in extending this tool or creating their own tool to interface with LEAPS. The second section describes the SAA layout and flow path. The third section describes the algorithm tuning settings available to a user of the SAA tool. The final section discusses some of the known limitations and major assumptions in the implementation of the SAA tool.

Programming Language selection and graphical user interface toolkit

Initial language selection was a very large part of the decision process when laying out the foundation for the SAA tool. Selection of programming language affects platform availability of the SAA tool (Windows, Linux, etc.). The choice also affects what capabilities are ultimately available to the program, such as Microsoft Excel integration. The choice most importantly affects the reach and applicability of the program into a particular user base. In the case of the SAA tool, the purpose was to extend the accessibility of the program to all users of ASSET for ship design.

The graphic user interface (GUI) toolkit selection has the same concerns as language selection. However, there are additional concerns for program design imposed by the GUI toolkit when modularizing or compartmentalizing source code. Model/View/Controller (MVC) software patterns are used to help sub divide program code into more manageable chunks. GUI selection affects the ease of MVC implementation.

The program language and presentation framework selection was an important part of design of the SAA tool. The next couple sections explain the thought process into the specific language choice of a C++/C# hybrid program and the Windows Presentation Format (WPF) for the GUI toolkit.

C++ / C# Hybrid programming language

A primary concern of the program is the ability to automatically import from an ASSET project source file into the SAA tool. There are too many naval architectural programs on the market today that rely on manual conversion from one software program to another. These manual conversions are usually cumbersome, time consuming, and prone to errors. These issues exacerbate as the ship project refines in later design stages and the design variables it takes to describe the ship grow in complexity. For these

reasons a direct automated conversion process was necessitated. The LEAPS application programming interface (API) is used in access ASSET source files (NAVSEA, 2005). The LEAPS API is implemented in C++ in the hopes of achieving platform independence in the future (LEAPS API as of now is limited to 32-bit windows programs). Therefore at least a portion of the SAA project was required to be in C++.

Besides the portion of the code to access a LEAPS database file, the rest of the program could be written in any other language with wrapping function calls. C# was selected as the chosen language for specific reasons. The first reason is primarily in the choice of the GUI toolkit used to interact with the user. Specific reasons for GUI toolkit selection are explained in detail below. Another major reason for C# was the issue of memory management. In C++, memory is allocated and managed directly by the programmer. Debugging issues not central to program implementation arise when the memory management is handled poorly. C# alleviates memory management concerns for the most part by using what is known as a garbage collector. A garbage collector handles creating and destroying dynamic objects in memory and takes this menial task away from the programmer so that they can focus on program implementation. Relieving some of the memory management concerns helped in the implementations of the three algorithms above, where deep copies of the compartment descriptions were required. Finally, C# was chosen over a Matlab implementation for the desire to be run without dependence on a separately installed program, even though Matlab handles memory management extremely well. The intent of the SAA tool implementation was to run on any machine that can currently run ASSET.

Windows Presentation Foundation (WPF) for presenting model views

A primary reason that WPF was used to implement the SAA tool was for the MVC software pattern or what Microsoft refers to as the Model/View/View-Model (MVVM) pattern (Sells, 2005). MVVM essentially separates GUI program flow into three modular sections known as the view, the view-model, and the model. The view is what the user interacts with on the screen. The view-model is the intermediary between the view and the model sending changes to the view that have occurred in the model or translates clicking interactions to manipulations on the model. The model is essentially where the data resides and where procedures such as the algorithms that are described above are performed.

WPF support for MVVM software pattern is very well developed. One example is in the form of Data-Binding a view to the model. When a model variable is Data-Bound to the view, any changes to the model variable are directly updated to the view. The programmer is responsible for directly changing the view when a model variable changes in other toolkits such as QT, HTML5, or Microsoft Foundation Classes (MFC). There are a minimum of eight hundred different variables alone displayed to the view to

describe SSCS area descriptions from ASSET. Data-binding to the view allowed me to concentrate on program implementation and not have to worry about micromanagement of the view.

One final reason for the selection of WPF is the ability to render two dimensional and three dimensional views to the screen. 2D and 3D rendering in C++ GUI toolkits are usually handled with direct function calls to graphics cards from DirectX or OpenGL API's. Both API's require the programmer to develop to a low level and setup the display view for use. For perspective, displaying a single colored triangular face with these APIs requires the programmer to write micro programs sent to the graphics card on how to draw the three vertices of the face, set up lighting to illuminate the view, and setup a viewport into the 3D space world. All these tasks are not trivial and distract from the implementation. There are some third party libraries that exist to alleviate these low level tasks, but they usually have issues with software licensing and program ownership. WPF, on the other hand, already has class objects in the toolkit to simplify setting up 2D and 3D views.

User interface layout

The user interface layout of the SAA tool implementation was constructed to reinforce the three stages proposed for general arrangements planning: volume allocation, one dimensional area assignments, and two dimensional area assignments. Although the flow path as described above is the best use of the tool, any stage can be performed independently of the other two stages in the SAA tool. The initial input of the program is assumed to be an ASSET ship that has had the FOCUS utility feature run on the project to prepare it for use in other LEAPS programs. The outputs of the program are one-dimensional spreadsheets describing the area placements in the ship compartments and a drawing exchange formatted (.DXF) file that can be imported into other CAD programs for detailed design work.

Volume Allocations View

Figure 8 shows a view of a sample project open to the volume allocations stage of the SAA tool. The main window pane shows a three dimensional view of the ship. This view can be used to select different compartments to operate on in the other areas of the tool. The color of the compartments map to their intended function such as the light blue color meaning the compartment is designated as an arrangeable compartment. The 3D perspective is useful to the designer by showing the global locations of other compartments with respect to each other. The 3D view supplements other views in the tool and allows the designer visualize assess how the ship is broken into different functions.

The top of the two panes to the right is a volume accounting spreadsheet for the ship that updates after compartment assignments are made. The bottom pane is where compartment

assignments to SSCS function are made as well as zonal assignments for the area optimization algorithms.

The menu bar at the top of the page displays a global accounting of space area and volume to the user. It also provides an access to the Zone editor to rename zones to something less arbitrary for the project.

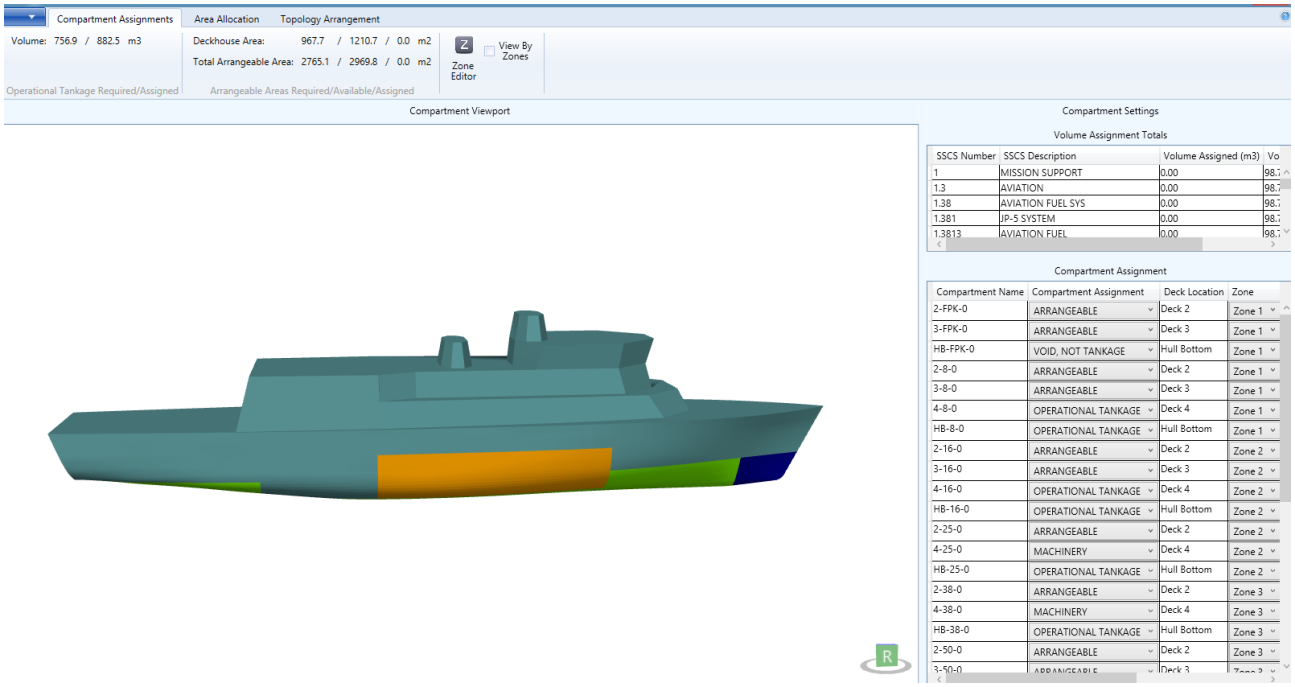

Compartment Assignments
Area Allocation
Topology Arrangement
Volume: 756.9 / 882.5 m3
Deckhouse Area: 967.7 / 1210.7 / 0.0 m2
Total Arrangeable Area: 2765.1 / 2969.8 / 0.0 m2
Zone Editor
View By Zones
Operational Tankage Required/Assigned
Arrangeable Areas Required/Available/Assigned
Compartment Viewport
Compartment Settings
Volume Assignment Totals
SSCS Number
SSCS Description
Volume Assigned (m3)
1 MISSION SUPPORT 0.00
1.3 AVIATION 0.00
1.38 AVIATION FUEL SYS 0.00
1.381 JP-5 SYSTEM 0.00
1.3813 AVIATION FUEL 0.00
Compartment Assignment
Compartment Name
Compartment Assignment
Deck Location
Zone
2-FPK-0 ARRANGEABLE Deck 2 Zone 1
3-FPK-0 ARRANGEABLE Deck 3 Zone 1
HB-FPK-0 VOID, NOT TANKAGE Hull Bottom Zone 1
2-8-0 ARRANGEABLE Deck 2 Zone 1
3-8-0 ARRANGEABLE Deck 3 Zone 1
4-8-0 OPERATIONAL TANKAGE Deck 4 Zone 1
HB-8-0 OPERATIONAL TANKAGE Hull Bottom Zone 1
2-16-0 ARRANGEABLE Deck 2 Zone 2
3-16-0 ARRANGEABLE Deck 3 Zone 2
4-16-0 OPERATIONAL TANKAGE Deck 4 Zone 2
HB-16-0 OPERATIONAL TANKAGE Hull Bottom Zone 2
2-25-0 ARRANGEABLE Deck 2 Zone 2
4-25-0 MACHINERY Deck 4 Zone 2
HB-25-0 OPERATIONAL TANKAGE Hull Bottom Zone 2
2-38-0 ARRANGEABLE Deck 2 Zone 3
4-38-0 MACHINERY Deck 4 Zone 3
HB-38-0 OPERATIONAL TANKAGE Hull Bottom Zone 3
2-50-0 ARRANGEABLE Deck 2 Zone 3

Figure 8: Volume Allocations View

Volume Assignment Output

The output from this stage are two spreadsheets accounting for the volumes associated with compartment assignments. The first spreadsheet tracks all the compartments on the ship and their respective assignment such as tankage or arrangeable space. The second spreadsheet tracks the SSCS space volume assignments against the required global values estimated by ASSET. For example, ASSET may set a required volume for tankage based on endurance range of the vessel. This required volume can be met by setting multiple compartments to tankage, so that the sum total of the tankage assignments meets or exceeds the required SSCS volume specified by the ASSET project file. Please see the attached appendices for an example of the SAA tool output.

One Dimensional Area Assignment View

Figure 9 shows a view of a sample project open to the one dimensional area assignments stage of the SAA tool. The window is divided into four panes representing four different views of the area and compartment information available to the viewer.

The top left pane is where the user sets his/her preferences for the required SSCS space areas to be placed on the ship. The user can select a zone preference, longitudinal location preference by percentage of LBP, deck location preference, as well as the ability to ignore area placement from the optimization algorithms. There is a command to load heuristic preferences that were compiled by surveying some naval ship designers on the general location preferences for some of the SSCS areas that exist on the ship.

The bottom left pane is where the user can fine-tune the utility function used in the optimization methods. The user has the ability to affect utility weighting, change the utility distribution functions, and select an averaging or minimum calculation method for utility composition function aggregation.

The top right pane is the SSCS area allocation accounting sheet similar to the volume allocation view spreadsheet in the first stage. The view also allows for the tracking of areas assigned specifically to the deckhouse compartments. Also, as in the volume allocation view, the spreadsheet dynamically changes to the area assignments made in the compartments.

The bottom right pane is a view of the area assignments of an individual compartment. This view is where manual adjustments to area allocation are made.

The menu bar as in the volume allocation view displays a global accounting of assigned area against SSCS required areas. The two optimization functions are called from this menu bar. Only one optimization method is necessary to be performed if the user chooses to perform automatic assignments.

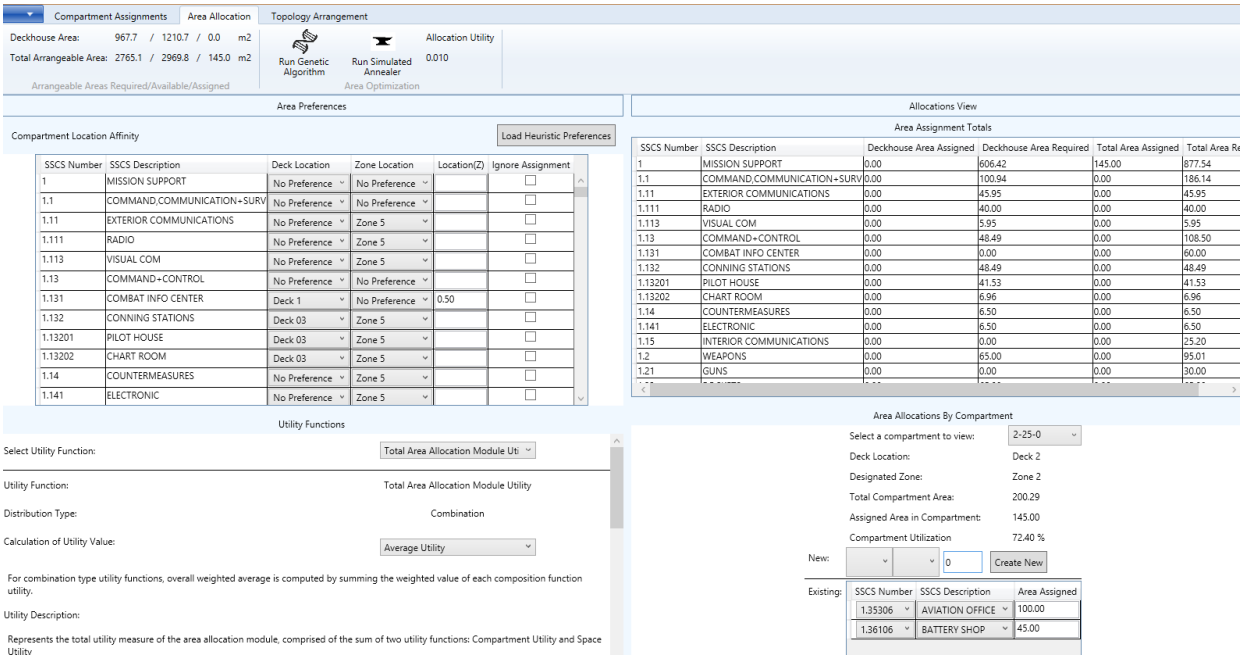


Figure 9: One dimensional assignments view

Area Assignment Output

The output from this stage are several spreadsheets accounting for the areas associated with space area assignments to compartments. The first spreadsheet tracks the SSCS space area assignments against the required global values estimated by ASSET. For example, ASSET may set a required area for crew berthing based on the hull size of the vessel. This required area can be met by setting multiple compartments to berthing, so that the sum total of the tankage assignments meets or exceeds the required SSCS area specified by the ASSET project file. The remaining spreadsheets of this stage are the area assignments by compartment sorted by deck level. Please see the attached appendices for an example of the SAA tool output.

Two Dimensional Area Layout View

Figure 10 shows a view of a sample project open to the two dimensional area layout stage of the SAA tool. This view provides the means to place and resize area assignments and set the passage templates for the ship.

The center pane on the right hand side is the main interaction is a two dimensional view of a deck on the ship. In this pane the user can use the camera controls to shift around the viewpoint and draw simple shapes onto the view representing assigned areas on the ship. Each compartment and area selected provide indication of their selection by a red enclosing bounding box around their borders. There is collision detection between areas from areas to the compartment bounding polygon and

between intakes and obstructions. Passageways do not provide collision detection to the user. This was intentionally omitted to allow the user to move assigned areas easily around the passageways.

The top left pane allow the user to manually set passage templates for individual compartments or automatically generate a passage layout for the ship based on passage width and DC deck selection. The user can select to automatically generate a single or a double passage layout.

The bottom left pane is an area accounting spreadsheet that displays what the user has assigned the selected compartment in the one dimensional spreadsheet and what has been physically laid down on the two dimensional view. Areas are labeled by their SSCS number and not there function, and there is a way to toggle the label off if desired.

The menu bar also displays a comparison between area assignments against area available. It also includes the button to start the greedy filling algorithm. Camera controls for the two dimensional view are also set in the menu bar. Two draft tools are available to be used in the two dimensional view. The first is a standard rectangle creator, and the second is a polygon creator. The polygon creator produces only polygons that are not self-intersecting and that do not enclose other areas.

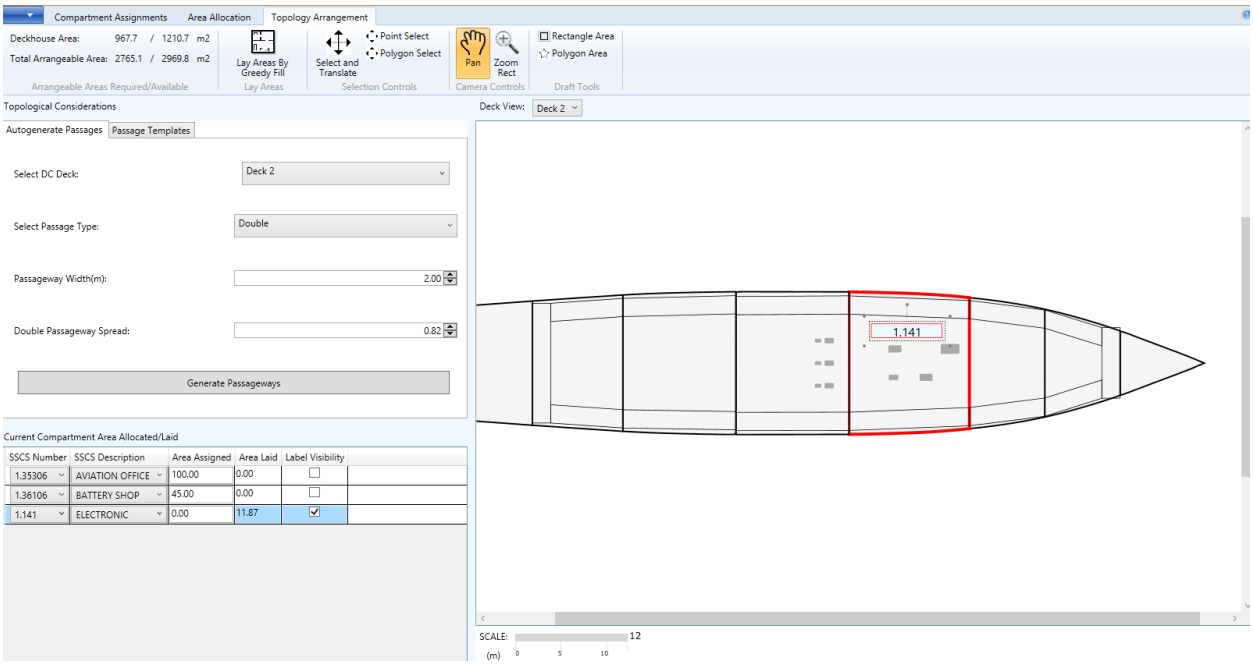


Figure 10: Two Dimensional Area Layout View

Area Layout Output

The output of this stage of the SAA tool is a drawing exchange file (*.dxf). This file includes the inboard profile of the vessel specifying the hull compartment assignments, and a layer for each deck level on the

vessel. This format was chosen as the output of this stage because of the universal acceptance of CAD programs to this file type (Autodesk, 2008).

Utility Function and Algorithm Settings

The utility function by default places equal precedence between compartment filling utility and space satisfaction utility. The compartment filling utility is set to a Gaussian Distribution with a mean of 80 percent desired filling rate and standard deviation falloffs set to 0.04 on the left side of the mean and 0.02 for the right side of the mean. The nominal 80 percent mean value is set to account for distributed systems and passageways and other design growth margins set by the designers. The falloff ranges are different from each other in the Gaussian utility distribution to emphasize that overfilling a compartment is worse than under filling the compartment. Zonal preference utility, deck preference utility, and longitudinal preference utility are equally weighted to comprise the space satisfaction utility. Gaussian distribution settings between the deck and longitudinal utility functions are set to the same values: 0 percent mean deviation as a percentage of deck height and LBP respectively, and 10 percent for the falloff in standard deviation on both sides of the mean. All utility functions are set to "average," meaning all utility values are calculated by the weighted sum of their individual pieces.

The genetic algorithm and simulated annealing commands have some settings built in that can be changed by the user to suit their preferences. Nominal values for these default settings were recovered from mock test cases run on sample ship problems. The genetic algorithm default settings are set to the following:

GENETIC ALGORITHM DEFAULT SETTINGS	
NUMBER OF ITERATIONS	2000
CHROMOSOME POPULATION SIZE	50
MUTATION RATE	0.007
WHEN TO RESEED	200 iterations with no change in best solution
NUMBER OF RESEEDS	5

Table 1: Genetic algorithm default settings

The number of iterations determine how long the algorithm will run. Chromosome population size determines how many chromosomes are in each generation of the algorithm. The mutation rate dictates how often mutation occurs per generation. The variables set for when to reseed and how many times are used to determine when the solution has reached convergence and when to stop the

algorithm early. These tuning variables were discussed in detail in the genetic algorithm section for more information.

The simulated annealing algorithm has two tuning parameters. The first parameter dictates how many iterations of simulated annealing are performed. By default this parameter is set to 15000 iterations. The second parameter is the cooling factor which is used to set the acceptance probability of less viable solutions for the simulated annealing algorithm as the process iterates. The cooling factor is not the acceptance probability but rather a factor applied in the acceptance probability calculation to properly scale its value with the number of iterations. This parameter is set to 0.003 by default.

Limitations

Noticeable limitations are present in the SAA tool program just like there are faults to be found in every program. Some of these limitations can be fixed by adjusting or adding to the SAA tool functionality. For example, adding support for stairwells could be accomplished by incorporating them into the passage template designs. Then the automatic passage generation tool could properly position the stairwells across the decks of a ship. Another example limitation that could be fixed by adding to the SAA tool is handling the reconfiguration of the intakes and exhausts. The intakes and exhausts are treated as a fixed obstruction that must be designed around. In actuality, as long the intakes and exhaust stacks are sized properly, the designer can reroute the stacks as needed within some limitations.

While zoning allows for some areas to be grouped together, there is nothing provided in the user interface to specify area averseness to each other. For example, it is undesirable to place hazardous flammable materials next to ordinance. The decision to leave out averseness preferences into the optimization was deliberate. There are over two hundred different SSCS numbers to account for and mapping out a dissociation matrix for SSCS categories against all the other SSCS categories would require a nominal two hundred by two hundred matrix or forty thousand entry preference matrix for input into the utility functions. While the optimization methods would be able to handle matrices of this size, time might be better spent manually assigning spaces to compartments than filling out an averseness/affinity preference matrix.

Unfortunately, some limitations might require a shift in thought process away from the three stages discussed above. One example of this kind of a limitation is being able to choose from different families of viable solutions. As of right now, only the optimal solution of the algorithm is saved and displayed to the user. There are several local optima that could all be potential solutions, but identifying

these solutions is not available in the SAA tool. Presenting and communicating the existence of these solutions is not provided for either, even in the event they could be determined. Another limitation is that the tool does not handle distributive systems well with the exception of passage layout. Chill water distribution and electrical distribution are examples of systems that are distributed throughout the ship. A shift in the planning process or the development of another tool may be required to address this concern.

Finally, some of the limitations are caused by the operating platform performance. The upper bound on the number of compartments the tool can import is around one hundred and fifty. This is due to a two gigabyte random access memory limit for 32 bit processes running in Windows. Each three dimensional model in WPF is fixed in size no matter how many vertex elements compose the model; small compartments take up the same size as large compartments. This limitation means ships that use longitudinal bulkheads as structural elements might have issues being imported into the SAA tool because more compartments are generated in those projects. Lastly, because the SAA tool was developed using WPF, the operation of the SAA tool is limited to Windows machines. This is not a problem as of now, but LEAPS is eventually going to be ported to Linux systems and this SAA tool will not be available on those systems without a redesign of the GUI.

4. Application Examples

This section is broken down into two major parts. The first part presents a test case that demonstrates the ability of the area allocation and area layout stages under a test case with immediately perceivable correct solutions. The second part of this section discusses the application of the SAA tool to a nominal test case from import of an ASSET ship database to the generation of CAD drawings.

Basic test case of algorithms

The following test case is presented as a proof of concept of the last two stages in the SAA tool against an “easy” test case. This test case has several global optimum solutions based on the set preferences; however, these solutions are much fewer in number than the number of infeasible solutions. The test case size was selected to be large and complex enough that solutions could not be recovered by chance with random number generation. However, the test case was small and simple enough that to saliently show that there are only a few subsets of correct global optimums.

Test Case Ship Description

The test case ship was a box barge concept with a total of ten compartments. There were two decks with a total of five compartments on each deck. Each compartment was sized at 220 area units with a

compartment height of 3 area units. The entire ship was sized to 2200 total area units. The ship was divided into three zones comprising of a forward zone encompassing the first two compartments on each deck, an aft zone encompassing the last two compartments on each deck, and finally a zone from the other two midship compartments. Figure 11 is provided to summarize the ship description.

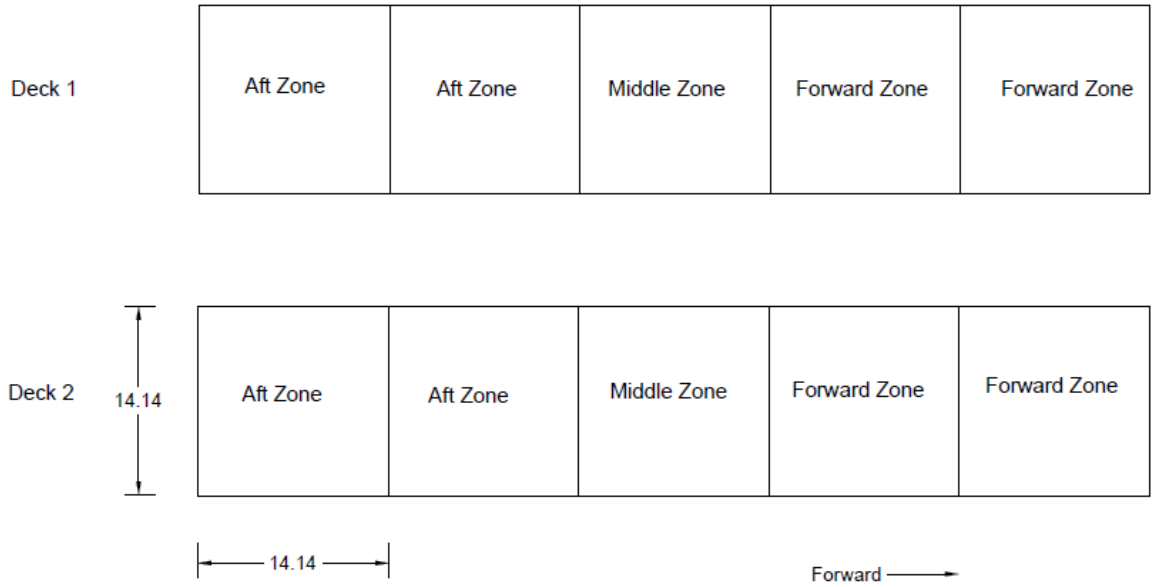


Figure 11: Test case ship zonal breakdown

There was no tankage on this test case ship which means all compartments were considered to be arrangeable. A set of twenty space area requirements were to be placed in the ship. Half of the required areas were sized at 50 area units and the other half of the required areas were sized at 150 area units.

Area Allocation Stage

The area preferences were set from the area requirements to have no conflicting preferences. The first of ten areas were set on the second deck, within this set the first four areas preferring to be set to the forward zone and the second four areas to be set in the aft zone, and the remaining two areas to be set at a ship longitudinal preference at 50 % of the total ship length. The other set of ten areas had their preferences set similarly to the first set but had a preference to be placed on the first deck. Table 2 shows these preferences in tabular format. The total area required by all space requirements on the ship was 2000 area units. The compartment filling mean function was set to fill at 91 % so that each compartment would need a 50 area unit space with a 150 area unit space to achieve optimal filling utility.

Area Number	Area Required	Deck Preference	Longitudinal Preference	Zone Preference
Area 1	50	Deck 2	None	Forward Zone
Area 2	150	Deck 2	None	Forward Zone
Area 3	50	Deck 2	None	Forward Zone
Area 4	150	Deck 2	None	Forward Zone
Area 5	50	Deck 2	None	Aft Zone
Area 6	150	Deck 2	None	Aft Zone
Area 7	50	Deck 2	None	Aft Zone
Area 8	150	Deck 2	None	Aft Zone
Area 9	50	Deck 2	0.50	None
Area 10	150	Deck 2	0.50	None
Area 11	50	Deck 1	None	Forward Zone
Area 12	150	Deck 1	None	Forward Zone
Area 13	50	Deck 1	None	Forward Zone
Area 14	150	Deck 1	None	Forward Zone
Area 15	50	Deck 1	None	Aft Zone
Area 16	150	Deck 1	None	Aft Zone
Area 17	50	Deck 1	None	Aft Zone
Area 18	150	Deck 1	None	Aft Zone
Area 19	50	Deck 1	0.50	None
Area 20	150	Deck 1	0.50	None

Table 2: Test Case Area Preference Table

The design space of the solution sets is 20^{10} or slightly over 10 trillion different possibilities that could result. Random chance or straight enumeration of the design space is not likely to find the global optimal solutions that exist. There are a few sets of the global optimal solutions that exist; for example, pairing Areas 1 and 4 in a forward deck 2 compartment or pairing Area 1 and 2 in a forward deck 2 compartment would both maximize the filling utility. The genetic algorithm implementation of the area optimizer was run several times, each time resulting in an overall utility of value of 1.0. Metaheuristic approaches use some form of randomization to search the design space. For the genetic algorithm, this randomization occurs in the chromosomal operations and initial chromosome seeding; as a result the

SAA optimization has a random chance to produce any one of the global optimal solutions. Please refer to Appendix A: Area Allocation Spreadsheet (Test Case) for one of the test case results in the SAA tool output spreadsheet form.

Two Dimensional Area Layout

The test ship case uses a single longitudinal 1 unit-wide passageway that runs from the forward extent to the aft extent of the box-shaped ship. This passage is present on both decks. Because this passage is centerline on the ship, it is impossible to leave the 150 area unit fully intact in each compartment. Each compartment therefore needs to place three areas, one 100 unit area and two 50 unit areas, after considering the passage. The output of the program is shown in Figure 12 and Figure 13.

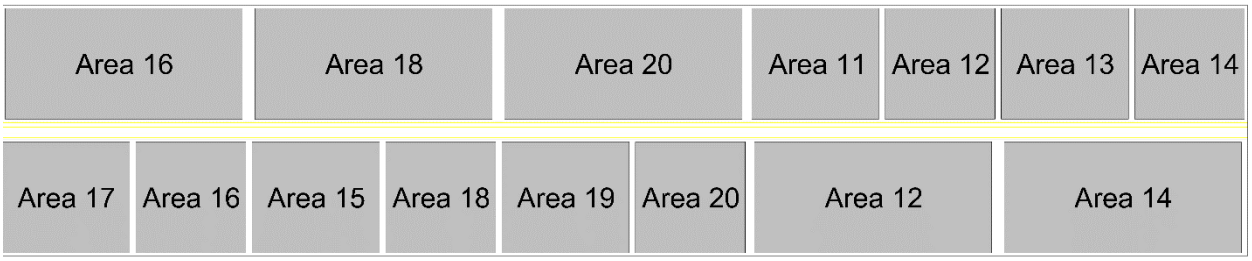


Figure 12: Deck 1 Arrangements

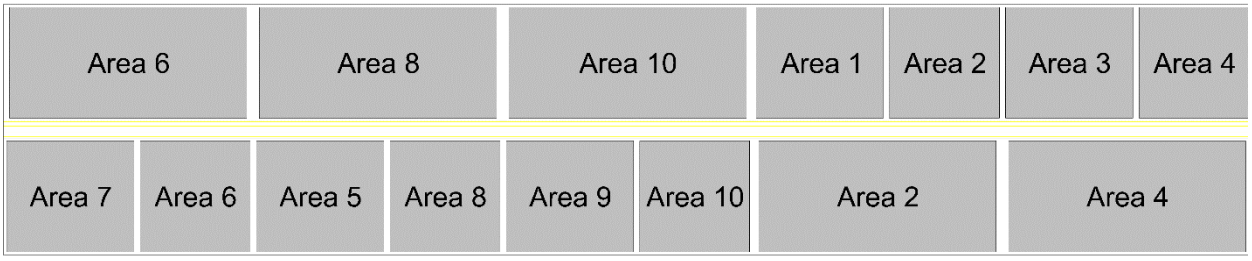


Figure 13: Deck 2 Arrangements

Application to an ASSET developed ship

A nominal ship was the constructed in ASSET to fully test the SAA tool flow path to determine its feasibility as a design tool. This section is subdivided into the three stages of the tool flow path as they are applied to the nominal design and any adjustments made to the project against the default settings. The final section shows the general arrangement two dimensional results before going into more detailed CAD programs.

Nominal Ship Description

The nominal ship was constructed in ASSET derived from a medium endurance coast guard cutter. Its overall length is 87 m, with a beam of 15.6 m, and a design waterline of 4.95 m. Its overall displacement is 3353 metric tonnes. The ship is manned with 120 individuals. The hull is subdivided by six internal decks and seven transverse bulkheads. Tankage requirements were derived from its endurance range and hull resistance. Area space requirements were derived from an arbitrary payload adjustment table applied to parametric space reservation values based on hull size and manning. The specifics behind the generation of area space requirements from payloads and parametric values is beyond the scope of this thesis. For a detailed spreadsheet of the required space areas please refer to Appendix B: ASSET Generated Area Reports.

Volume Allocations

Compartment assignments were originally assigned during the intact stability and strength analysis in ASSET. The compartment assignments were set to match those project settings. Aviation fuel tankage was placed below the aviation hangar. Operational tankage was set low in the ship around the engine room. The volume required to be set for operational tankage based on endurance requirements was 780 m^3, and 883 m^3 of volume was assigned.

The default zone arrangement suggested by the SAA tool was used for the ship's general arrangement assignments. The ship was divided into five zones with the ship hull divided into four equal zonal slices and the deckhouse structure set as the fifth zone. In total there are 33 compartments to the ship, of which 20 compartments were selected as arrangeable spaces. For a detailed excel report output on the specific compartment volume assignments please refer to Appendix C: Area Allocation Spreadsheet (Applied Ship Case).

One Dimensional Settings

The Gaussian mean for the filling function was not adjusted from its default value of 80 percent after accounting for margin in access passageways and distributed systems. The ship required area to be placed totaled 2712 m^2 and the total area deck available to the ship totals 2761 m^2. The heuristic preference command was used to generate longitudinal, deck and zone preferences for the SSCS required areas. The aviation hangar, pilothouse, and anchor handling spaces were fixed to specific compartments prior to running the genetic algorithm space optimization. These fixed placements account for approximately 10 percent of the required areas before the genetic algorithm places the remaining areas. The genetic algorithm was run under its default settings.

The resulting algorithm output resulted in an overall utility value of 79.3 percent. Near unity utility, indicating maximum satisfaction, cannot be achieved if area placements conflict with each other or cannot be placed in their desired location due to compartment filling. One example of this compromise is the manual placement of the anchor handling equipment at the foc'sle. The foc'sle compartment is sized at 23.76 m2, however, the anchor handling space is sized at 22.01 m2. This results in this compartment's filling to 92.62 percent, which in turn lowers compartment filling utility, in order to achieve the optimal area preference satisfaction. A remedy to this conflict of interest could be to introduce local compartment-filling utilities instead of the global compartment filling value of 80 percent. This is an area for future work on the SAA tool and process.

Unfortunately, the SAA tool does not give a breakdown of the satisfaction level for each utility measure composing the overall utility. The SAA tool does not explicitly state the compartment filling utility or the individual area preference utilities. The reason for this is to allow for process modification in the future where a new utility function could be used such as using local compartment filling preferences instead of a global filling preference described earlier.

Some areas had to be manually adjusted following the optimization. One of these areas was aviation handling which the algorithm wanted to place at about 40 percent of the ship length. Aviation handling space was relocated under the helicopter hangar. In a future reiteration of the design, the ship hangar should be resized to fit all the aviation equipment for helicopter support. The aviation handling space was replaced by galley spaces which shifted the galley spaces from under the helicopter hangar more forward and under the deckhouse. Another change was to manually split the reconfigurable special missions space into a large area placed on deck 2 and a smaller area on deck 1. This change was performed to provide better usage of the special mission's space on an operational level. These changes conflicted with both the compartment filling and the default heuristic preferences table. The resulting utility value after these changes was 72.8 percent, which means there can be improvements to the utility function calculations or preferences to define a better ship to optimize. For a detailed excel report output on the specific compartment area assignments please refer to Appendix C: Area Allocation Spreadsheet (Applied Ship Case).

Two Dimensional Settings

A double passageway was used to generate the passage layout. The standard passage width is at 2 meter spacing. The DC deck was located on Deck 2. Deck 2 is the first continuous deck running the length of the ship above the machinery and auxiliary machinery rooms. Each compartment below the DC deck was split with a zero width single passage template. The reason for setting a zero width passage

was to help with some of the dimension ratios of space areas in the compartment because of the way the greedy filling algorithm works.

Stairwells were drawn manually prior to running the greedy filling algorithm. Dimensions of the stairwells were 2 m by 4 m. There are four stairwell locations on the ship. One stairwell was placed at the bow and stern ring sections of the passage layout, and the other stairwells are above the engine room amidships to provide a central vertical access to the ship. The more structures or gridded a compartment is by obstructions the better the visual two dimensional output becomes.

The results of the greedy filling algorithm were exported to Rhinoceros 5 for detailed CAD arrangements. The results of the greedy filling algorithm were not adjusted, thus showing the capability of the algorithm to lay areas without operator intervention.

Results

Figure 14 through Figure 20 are images generated from Rhinoceros 5 of the SAA tool output. An inboard profile is presented first to show the compartment assignments that were made in the SAA tool during the volume allocation stage. The general arrangements for each deck are then shown. The output of the SAA tool was not adjusted, thus showing its raw output. Obviously, a touch-up is required after using this program to produce more accurate drawings. Annoyances like small aspect ratio areas and labels can be resized or repositioned to produce a more presentable output. The drawings are not meant to be viewed at the scale of 8 ½ by 11 inch paper; the inclusion of the drawings herein are to show the types of shape output of the two-dimension greedy fill algorithm. Dark grey areas are the SSCS areas that were placed by the algorithm. Yellow highlighted areas are the passageways and stairwells set by the passageway generator. Cross-hatched grey area outlines seen in Figure 17 through Figure 20 indicate intake and exhaust stacks. Please refer to Appendix D: Ship Arrangement Drawings (Applied Ship Case, for the drawings scaled to landscape orientation on 8 ½ by 11 paper.

Inboard Profile

An output of the volume allocations stage is the compartment inboard profile of Figure 14. Specific area designations of each compartment are not set in this profile. Arrangeable and deckhouse compartments will require labeling in the CAD drawing for more detail. Specifics to note are the locations of the operational tankage in the profile and the amount of tankage required to meet endurance range requirements. The machinery spaces are the two large compartments amidships, and operational tankage lies on the hull bottom of the vessel. Aviation fuel tankage and ballast are assigned to the aft

two compartments on the hull bottom. There is one arrangeable compartment on Deck 4 that is forward of the machinery spaces. Hull voids are assigned at the bow of the vessel.

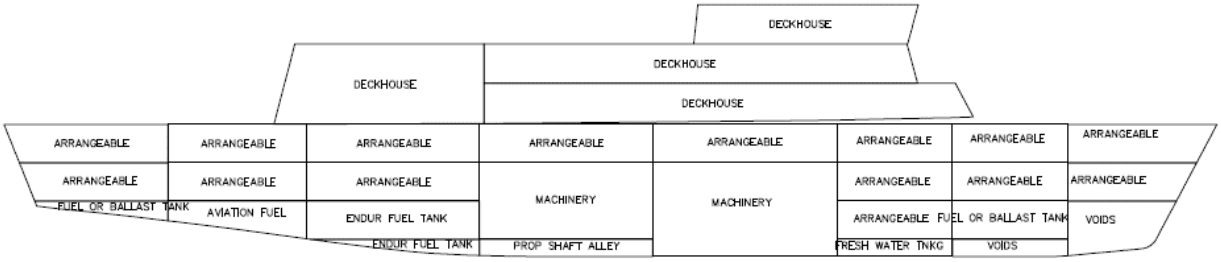


Figure 14: Inboard Profile

Deck 4 arrangements

Deck 4 arrangements are shown in Figure 15. Deck 4 only has one arrangeable compartment on its level with the rest of the compartments being engineering space or operational tankage. Assigned SSCS areas to this compartment are ship storage spaces and 400 Hz electric distribution area.

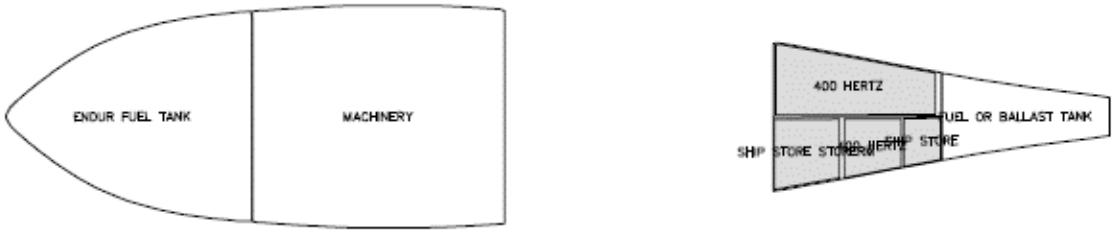


Figure 15: Deck 4 Arrangements

Deck 3 arrangements

Deck 3 arrangements are shown in Figure 16. Berthing and stowage areas were preferably located on this deck by the genetic algorithm. The first instances of areas with small aspect ratio areas are shown here. These areas can be resized in the SAA tool or resized in other CAD programs.

Figure 16: Deck 3 Arrangements

Deck 2 arrangements

Deck 2 arrangements are shown in Figure 17. Deck 2 was designated as the DC deck. A double passageway starting and ending in a ring formation is set for the access arrangements for the ship. Interesting aspects of this arrangement are that ship office spaces are located near the fantail, the galley and its supporting equipment are located forward of the superstructure, and the aviation supporting spaces are located under the aircraft hangar. Special missions space fills an entire compartment just forward of the aviations support spaces.

Figure 17: Deck 2 Arrangements

Deck 1 arrangements

Deck 1 arrangements are shown in Figure 18. Deck 1 is the first level of the deckhouse. The aft compartment is for aircraft stowage in the helicopter hangar. The forward compartment has additional stowage for the rest of the required stowage area for aircrafts. Internal deck area assigned to Guns is also set in this space as well as the general living space for the ship. Most of the areas in the forward compartment on this deck will require a manual resize. At the very least, the footprint size of these areas requiring adjustment are a better indicator of actual size than a spreadsheet one dimensional quantity.

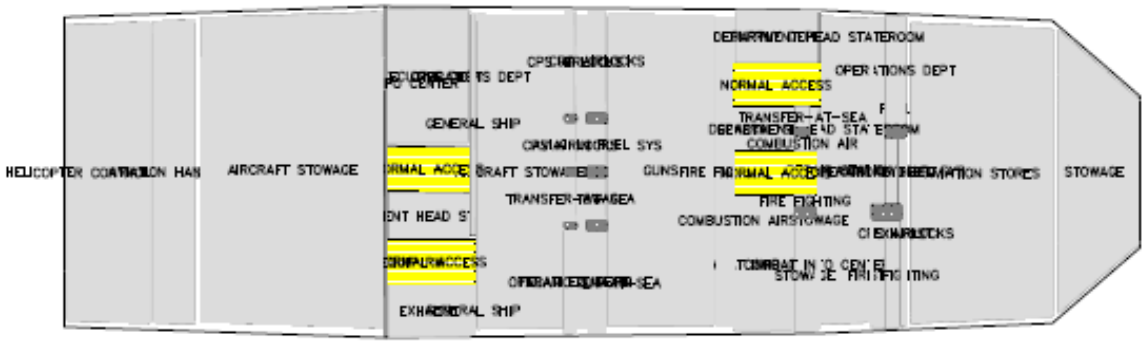


Figure 18: Deck 1 Arrangements

Deck 01 arrangements

Deck 01 arrangements are shown in Figure 19. An interesting aspect of the arrangement is the location of the one of the main fan rooms for ventilation of the ship is located on this level. The CO stateroom is on this level but will require a manual adjustment for the right shape for a stateroom. This deck will require modification to many of its areas to be useable due to the aspect ratios of the spaces.

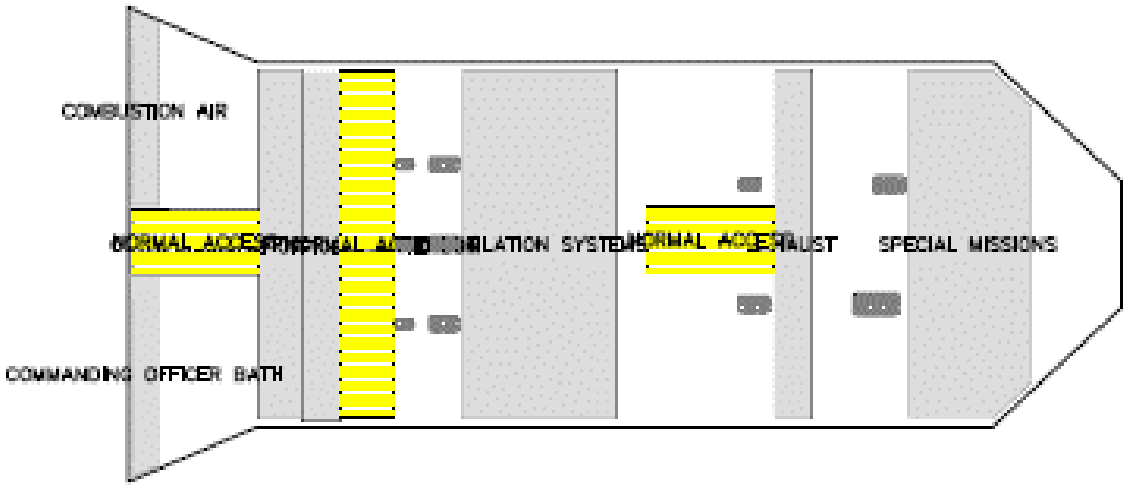


Figure 19: Deck 01 Arrangements

Deck 02 arrangements

Deck 02 arrangements are shown in Figure 20. Deck 02 consists of a single compartment and is the highest deck in the deckhouse structure. Interesting aspects of this arrangement are that the pilot house is located forward in the structure, aviation control equipment is located aft in the main structure, and chaff-countermeasures control and weapons are also in this space.

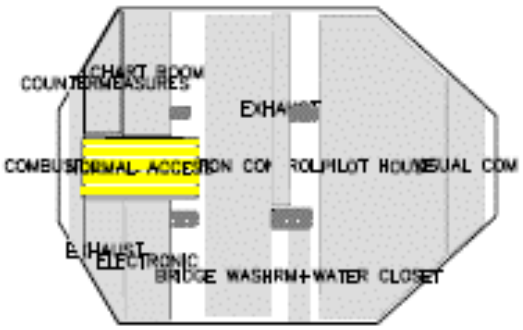


Figure 20: Deck 02 Arrangements

5. Summary

Conclusions

As shown by the the two case studies above, the SAA tool is powerful and can help translate the one dimensional SSCS volume and space area requirements into a starting design for detailed arrangements in more advanced CAD programs. The SAA tool and its design approach have qualities that make it very beneficial to ships that are both large and small. Perhaps the best aspect of the SAA tool is that it aids the user in the visualization of spatial quantities that are presented in one dimension in ASSET.

General arrangements are not finished at the end of the two dimensional layout stage. Electrical, thermal, or firefighting and other distributed systems remain to be addressed in the general arrangements. Topside arrangements also need to be considered. If not already addressed, topside arrangements can affect general arrangements below decks. Also, once major ship components are known, the detailed design of each area reservation should be updated to include those ship components. However, the large list of the remaining items to accomplish is significantly lessened by the use of the SAA tool and design approach.

Opportunities for Future Research

This project can also a stepping stone for other 2N design projects and theses, because it provides a framework for integrating program information and LEAPS databases with the C# wrapper library. There is no question that LEAPS database format is the direction that all professional naval ship design solutions must accommodate to stay relevant for U.S. Navy design interest. In this tumultuous time of shrinking design budgets, there is little to no incentive to convert projects that have been constructed in Matlab or other academic implementations that do not already accommodate the LEAPS structure. The framework C# library developed in conjunction with this project can be implemented in other design projects that are completely separate from this project. For example, the information from an ASSET ship project can be augmented by a new thermal or electrical distribution design tool. The possibilities to extend an ASSET ship project with 2N design tools are endless but an effort should be made to talk the language of ASSET and LEAPS from the start of development to remain relevant. Opportunities for future research were discussed as they became relevant in the previous sections. Some of the opportunities are based off extensions of this research and some would require a shift in thought away from the three stage process.

As mentioned in the implementation section above, there is a multitude of ways the SAA tool could be improved. The most basic of which is to upgrade the already existing functionality. Some

examples of such upgrades are being able to assign more than one zone to a compartment, fine tuning the heuristic preference load function with better starting preferences, and upgrading the passage templates to include stairwells. As of now, only one zone can exist per compartment; however, it is conceivable to have zones that overlap such as a berthing zone and an electrical distribution zone. The heuristic starting preferences were taken from a survey of my classmates and they could be better adjusted from parametric data from historical designs. Finally, the passage templates could be upgraded to include stairwells in the next tool iteration. These new templates then could also extend the automatic generation of passages to include stairwells as well. All of these functions are already present in the SAA tool, but further adjustments could be made to make these functions better.

Finally there are other research opportunities that exist outside of the three stage approach that are ripe for exploration. Andrews' building blocks approach is very intriguing because we have so many working designs to choose from already. If we could discretize the individual components or room configurations into useable pieces for a LEAPS catalog, there might not be a need for optimization techniques. The user could just select the 40 MW engine room that works and then make small adjustments that pertain to the individual design. For this project the building blocks approach was not as viable as the one-dimensional assignment optimization because of the lack of a LEAPS database catalog of these configurations. A great research opportunity exists in the construction of such a database. Lastly, in my research and construction of the SAA tool, academia and the professional world have made little progress in tools to handle distributed systems on the ship. I believe there is a lot of potential in the development of a design approach for general arrangements that handles these types of systems.

In closing, general arrangements is a large field with great potential for more research. The process of creating a good ship is often tedious. Research into general arrangements design approaches show that it does not need to be this way. Naval designers and architects can generate a good ship quickly and with little effort with the right tools, so that they can spend more time making the ship truly great. Some effort just needs to be spent to develop these tools.

References

Andrews, D. (2006, November 8). Simulation and the design building block approach in the design of ships and other complex systems. *Proceedings of the Royal Society*.

Anthony Daniels, F. T. (2010). Intelligent Ship Arrangement (ISA) Passage Variable Lattice Network Studies and Results. *ASNE Day Proceedings.* Michigan: University of Michigan.

Autodesk. (2008). DXF Reference. Mill Valley, California, USA.

D. Frey, P. H. (2009). The Pugh Controlled Convergence method: model-based evaluation and implications for design theory. In *Research in Engineering Design* (pp. 41-58). Springer London.

Denning, P. (1968). *Thrashing: Its Causes and Prevention.* Princeton: Princeton University.

Ginneken, O. (1989). *The Annealing Algorithm.* Boston: Kluwer Academic Publishers.

Luke, S. (2013, June). *Essentials of Metaheuristics*. Retrieved from Lulu: http://cs.gmu.edu/~sean/book/metaheuristics/

NAVSEA. (1984, March 23). Ship Space Classification System. *DOD-STD-2150*. Washington, District of Columbia, USA: Military Standard.

NAVSEA. (2005). LEAPS Application Programmer's Interface User Manual. Carderock, MD, U.S.: NSWC Carderock.

Nick, E. (2008). *Fuzzy Optimal Allocation and Arrangement of Spaces in Naval Surface Ship Design.* University of Michigan.

Oers, B. V. (2011). *A Packing Approach for the Early Stage Design of Service Vessels.* Netherlands: VSSD.

Olson, C. (1998). *Towards the Development of an Automated Ship Arrangement Design Tool.* Memorial University of Newfoundland: National Library of Canada.

Sells, C. (2005). *Programming Windows Presentation Foundation.* O'Reilly.

Appendix A: Area Allocation Spreadsheet (Test Case)

Appendix B: ASSET Generated Area Reports

Appendix C: Area Allocation Spreadsheet (Applied Ship Case)

Appendix D: Ship Arrangement Drawings (Applied Ship Case)